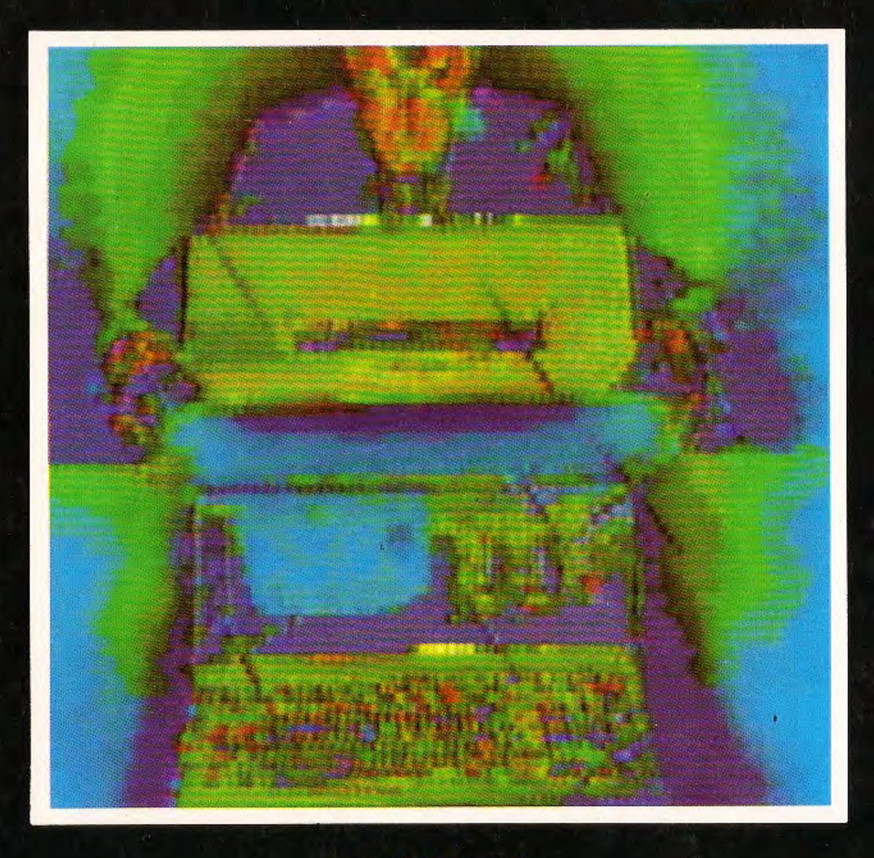
BIBLIOTECA BASICA INFORMATICA

SISTEMA OPERATIVO CP/M

11



INGELEK

BIBLIOTECA BASICA INFORMATICA

SISTEMA OPERATIVO CP/M

11

INGELEK

Director editor:
Antonio M. Ferrer Abelló.

Director de producción:
Vicente Robles.

Coordinador y supervisión técnica:
Enrique Monsalve.

Colaboradores:
Angel Segado.
Patricia Mordini.
Margarita Caffaratto.
Marina Caffaratto.
Francisco Ruiz.
Jorge Juan Monsalve.
Beatriz Tercero.
Fernando Ruiz.
Casimiro Zaragoza.

Diseño:
Bravo/Lofish.

Dibujos:
José Ochoa.

ISBN del tomo: 84-85831-45-4
ISBN de la obra: 84-85831-31-4
Fotocomposición: Pérez Díaz, S. A.
Imprime: Héroes, S. A.
Depósito Legal: M. 1329-1986

INDICE

PROLOGO

Hoy en día muchas personas afirman que la rápida "caída en desgracia" de algunos ordenadores domésticos y personales de la primera generación está íntimamente ligada a la pobreza de su sistema operativo, indigno de este nombre, y que, al limitarse a funciones elementalísimas, ha hecho el manejo del ordenador extremadamente incómodo al operador.

Bajo este punto de vista es bastante instructivo, además de útil, aprender la estructura, comandos y modalidades de funcionamiento del primer sistema operativo profesional de la historia de los ordenadores personales: el CP/M, desarrollado por uno de los genios del Silicon Valley. Hablamos de un personaje al que no le gusta figurar: Gary Kildall, el cual no sólo ha fundado la Digital Research (hoy en día una de las mayores empresas productoras de software de base y de aplicación), sino que también ha obtenido, gracias a una difusión "liberal" de su propia creación, la máxima popularidad por el CP/M.

En el campo de los microordenadores de 8 bits el CP/M costaba poco, estaba óptimamente documentado y tenía capacidad de adaptación a las más variadas configuraciones del hardware. Tres puntos que le han llevado al éxito, hasta tal punto de otorgarle el nombre de "bus de software". De hecho, la popularidad del CP/M en todas las máquinas dotadas de microprocesadores 8080 ó Z80 ha incitado a las casas de software a desarrollar, bajo su cómoda tutela, los programas de aplicación más variados, fácilmente transportables todos de un micro a otro.

Dentro del sector de los ordenadores personales de 16 bits no domina, sin embargo, el derivado directo de la primera criatu-

ra de Kildall, el CP/M 86, sino el MS-DOS de la rival Microsoft. A pesar de esto, hay dos hechos muy claros:

- la gran plataforma que suponen los micros profesionales de 8 bits (hay que tener en cuenta que también en el popular Apple II funciona el CP/M añadiendo una tarjeta de expansión);
- la validez del planteamiento del sistema.

El mismo MS-DOS, como se podrá constatar en el siguiente volumen de la BBI, dedicado a él, ha tenido en cierto modo en cuenta los aspectos fundamentales del CP/M; éste, por lo tanto, puede ser considerado un sistema que ha abierto brecha en uso profesional (y privado) del ordenador personal.

CAPITULO I

¿POR QUÉ UN SISTEMA OPERATIVO?

Dialogando con el ordenador

En este capítulo vamos a exponer los principales conceptos sobre la razón, significado e importancia de esa parte del "software" de base que se llama Sistema Operativo. Junto a otras herramientas (tools), entre las que figuran, en primer lugar, los lenguajes de programación y los intérpretes y compiladores, el Sistema Operativo entra también en lo que se llama "software de base", tal y como se vio en el anterior volumen de la BBI "Sistemas operativos y software de base".

¡Esperemos que sirva!

La comunicación con un ordenador presenta las mismas dificultades a las que una persona debe hacer frente al iniciar una gestión burocrática en un país extranjero sin conocer el idioma.

No es difícil imaginar tanto el asombro de los empleados de la oficina como el esfuerzo del extranjero en el vano intento de hacerse comprender mediante sonidos articulados, recíprocamente incomprensibles, y mímicas igualmente desconocidas.

Pero si de repente interviene una guía experta, se puede notar cómo la comprensión proyecta su suave y protectora sombra sobre la gris habitación de la oficina, la sonrisa ilumina las caras de los presentes, los papeles se mueven con precisión y velocidad de una mesa a otra, las incomprensiones desaparecen, las funciones se realizan con competencia y al final el sujeto sale de la oficina con la instancia rellenada con las oportunas firmas y sellos, la cara iluminada de satisfacción.

Si tuviésemos que hablar con un ordenador en su propio lenguaje, nos encontraríamos en la misma desesperada situación.

El manejo de un ordenador no consiste única y exclusivamente en conocer las órdenes a las que obedece, tarea ciertamente posible, sino también, y sobre todo, en el dominio del "cómo" estas órdenes, en su organización física y lógica, se lleven a cabo.

Basta pensar en la dificultad que llevaría consigo la conexión entre un ordenador y una impresora si cada vez que se realizase hubiera que especificar a la unidad de control del ordenador los niveles de las señales que debe respetar, los protocolos de comunicación, las señales de respuesta a examinar y transformar, etc.

También ocurrirá lo mismo para los problemas de conexión con una unidad de control de discos si tuviésemos que dirigir el tráfico de datos de entrada y salida y su organización sobre el disco.

Ahora bien, esa providencial "guía experta", en este caso para el mundo del ordenador, existe, desarrolla perfectamente su tarea y, dada su compleja estructura y actividad, merece el nombre de Sistema Operativo (abreviadamente, S.O.).

Estructura de un ordenador

Como ya sabemos por los anteriores libros de la colección, el problema de la estructura física y el funcionamiento de un ordenador compete más bien al campo de la electrónica. El fenómeno que hace posible la existencia del ordenador es la transmisión, codificación e interpretación de una señal elemental, que en este caso particular puede asumir dos estados (activo/no activo, +/-, positivo/negativo, 1/0, o cualquier otra definición capaz de individualizar dos situaciones opuestas).

Ya que, por lo menos desde el punto de vista lógico-funcional, la señal elemental presenta sólo dos estados, se ha desarrollado una completa estructura lógica para sistematizar el modo con el que la señal binaria debe ser manipulada, interpretada, codificada; en una palabra: utilizada para representar datos.

Nacen en este ámbito las definiciones de bit, nibble, Byte y Word, estos tres últimos múltiplos de la unidad elemental (el bit=BInary digiT, dígito binario).

Un ordenador está constituido por un conjunto de unidades lógicas destinadas a tareas específicas; en particular, las funciones que hay que desarrollar son: transformación y tratamiento de la señal binaria bajo la forma de datos, memorización temporal de los datos necesarios y todas aquellas que controlan que cada una de las operaciones requeridas se realicen en la secuencia y forma adecuadas y con las conexiones oportunas.

Para el uso práctico del ordenador son necesarios al menos dos accesorios: el teclado, que permite al usuario enviar sus órdenes a la unidad central, y la pantalla, que visualiza las órdenes transmitidas y las respuestas del ordenador.

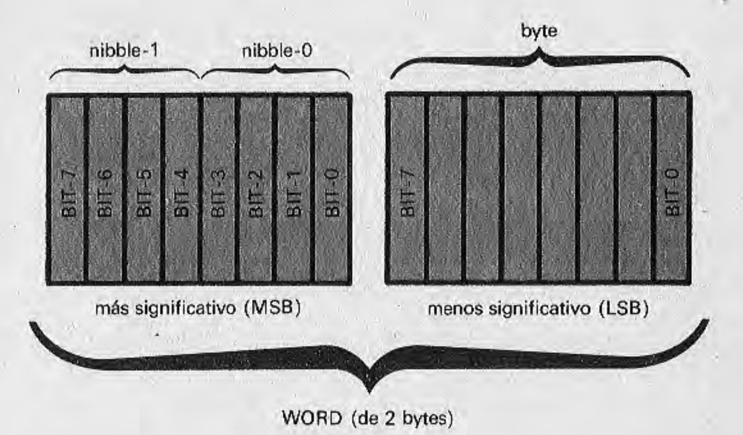


Figura 1.— Unidades de información más usadas, obtenidas todas a partir de la célula básica, el bit.

Se pueden añadir otras, como la impresora (para obtener la respuesta del ordenador sobre papel), las unidades de discos o cintas (para aumentar la capacidad de almacenamiento y la permanencia de los datos) y otras unidades de entrada o salida de datos; son los llamados periféricos.

Tareas de un S.O.

Volvamos al ejemplo del turista y el guía experto.

El S.O. es el que se encarga de satisfacer las necesidades expresadas por el usuario. De hecho, se ocupa de la gestión y control de todos los procesos que implican a las unidades centrales y a las periféricas, debiéndose preocupar el usuario tan sólo de especificar sus órdenes manteniendo una correcta sintaxis. Prácticamente, el S.O. oculta al usuario la estructura física interna del ordenador y su organización lógica, permitiéndole comunicarse con él a un nivel lo más cercano posible al modo humano.

Por ejemplo, si queremos conocer el tipo de información contenida en un disco basta con que pulsemos la orden convenida (DIR) y el sistema operativo se ocupa de:

- comprobar la sintaxis y congruencia de la orden;

- llevar a cabo las operaciones necesarias para leer en el disco la lista de los datos contenidos;
- comunicársela al usuario visualizándola en la pantalla o imprimiéndola sobre un papel.

Para comprobar o modificar la manera en la que se realiza una conexión con una unidad periférica, siempre será el S.O. el que se ocupe de realizar todas las complejas tareas que permitan que la orden se lleve a cabo, dejando libre al usuario para concentrarse plenamente con el fin de explotar al máximo las posibilidades que tiene a su disposición.

Naturalmente, la calidad y flexibilidad con que estas tareas se llevan a cabo varían de un sistema operativo a otro y de una computadora a otra, aunque mantengan inalterada, sin embargo, la naturaleza de los servicios ofrecidos y la idea de evitar al usuario la necesidad del conocimiento físico y estructural del ordenador con el que trabaja.

Cuándo interviene el S.O.

Ya que nuestro buen amigo el S.O. ha sido proyectado para ayudar al usuario en la comunicación con el sistema, es preferible que intervenga lo más pronto posible, incluso desde el mismo momento de encender el ordenador, pues ya entonces toma el control de la situación del ordenador y de todos sus periféricos. Existen diversos tipos de S.O., adecuados para distintos microprocesadores. Los ordenadores más modernos obran de manera especial, permitiendo la utilización de más de un sistema operativo.

Son también cada vez más normales los ordenadores que, en el momento del encendido, realizan una rápida y completa autodiagnosis de su estado. Cuando la "sesión" de autodiagnosis tiene buen resultado, el usuario tiene la posibilidad de introducir desde un disco un S.O. cualquiera de los que acepta el ordenador.

Esta función es desarrollada por un programa secundario particular, integrante en realidad del S.O., conocido como bootstrap, que asume la responsabilidad de completar el trabajo de trasladar a la memoria la parte restante (almacenada en disco) del mismo S.O. Es interesante observar cómo es el propio S.O. el que lleva a cabo todas las funciones necesarias para su propia carga.

Este particular parece querer ponerse en evidencia desde el propio nombre de la operación: Bootstrap (cordones de las botas) que procede de un dicho anglosajón que indica el hecho de lograr levantarse tirando de los cordones de sus propias botas.

En cuanto se ha efectuado la carga el S.O. se activa y toma el control de todas las sucesivas operaciones.

Es posible, por tanto, trabajar en el mismo ordenador con varios sistemas operativos. Sin embargo, el trabajo que se ha llevado a cabo con la asistencia de uno de ellos no se puede utilizar (salvo casos y condiciones particualres) cuando se pasa a otro S.O. porque no son compatibles entre ellos.

CAPITULO II

CP/M: EL BUS DE SOFTWARE

Introducción al CP/M

El CP/M (acrónimo de Control Program for Microprocessors) es un S.O. que actúa en ordenadores que utilizan discos flexibles (disquetes o floppy disk) o rígidos (hard disk) como soportes de memoria de masa en los que se registran con consistencia datos en cantidades muy superiores a las permitidas por la memoria central del ordenador.

Su principal característica es la facilidad con la que puede ser instalado en computadoras que utilicen como microprocesadores el Intel 8080 o el Zilog Z80, que tengan por lo menos 20 Kbytes de memoria central y que controlen no más de 16 unidades de disco.

De hecho, la transformación de una versión del CP/M en otra apta para otro ordenador requiere sólo la modificación de la sección específica que enlaza con la estructura física particular del ordenador que la recibe.

Las restantes secciones, es decir, las que realizan el interface con el usuario, son utilizables prácticamente sin modificaciones.

Là extraordinaria difusión del CP/M como S.O. para los ordenadores personales, sobre todo en el campo profesional, es debida principalmente a esta gran flexibilidad de instalación.

El CP/M es válido para controlar microordenadores con posibilidad de manejar los bits de 8 en 8. Esta dimensión del bus de datos de la computadora determina las características técnicas de las funciones proporcionadas. Existen también versiones del CP/M que controlan 16 bits (el CP/M-86, para el microprocesador Intel 8086), pero las funciones proporcionadas son semejan-

tes. La versión más moderna (la 3.1) es bastante revolucionaria con respecto a las anteriores, pero su escasa difusión y la competencia con otros sistemas operativos para 16 bits (como el MS-DOS) obligan a mostrarse mucho más cautos en cuanto a que se repita el éxito obtenido por el CP/M en su versión de 8 bits.

El CP/M ofrece la posibilidad de controlar datos e informaciones en la misma forma en que los usuarios los han generado o bien de acuerdo con el método de la misma computadora, que lo transforma por comprensibles motivos de velocidad.

El S.O. se ocupa del movimiento de estos datos, de la optimización del espacio que ocupan en el disco, de su catalogación, medida y control.

En el "ambiente" CP/M existen también diversos instrumentos para crear y corregir textos y ensamblar programas escritos con lenguaje de programación. Tiene disponible, de todas maneras, un amplio espectro de programas de aplicación al que los usuarios mismos y las casas de software especializadas aportan continuamente novedades. Esta gran cantidad de programas, con posibilidad de ser trasladados muy fácilmente a distinto hardware, ha originado el sobrenombre de "Bus de Software" al CP/M.

Unas pocas definiciones

El CP/M se presenta al usuario con un conjunto de comandos adecuados para crear, modificar, borrar, imprimir, combinar y trasladar datos a un disco. Los datos están representados en el ordenador por adecuadas secuencias de bytes, que corresponden a los caracteres contenidos en una tabla aceptada internacionalmente: la tabla ASCII (American Standard Code for Information Interchange). A este código ASCII pertenecen todas las letras, mayúsculas y minúsculas, del abecedario inglés, las diez cifras árabes, todos los signos de puntuación normalmente utilizados y caracteres especiales de control (ver el Apéndice A).

El múltiplo más utilizado para el byte es el K (Kilobyte), que corresponde a 1024 bytes, es decir: 2^{10} bytes.

Mediante uno o más bytes representamos valores alfanuméricos (cada carácter representado por un byte), numéricos (de distintas características y representación: entero, real en simple precisión...) y comandos especiales en forma de caracteres de control (ej.: retorno de carro, nueva línea, señal de texto terminado, etc.). Cualquier tipo de información se codifica en uno o más bytes.

Sobre el disco los bytes están organizados en estructuras homogéneas, cada una de las cuales constituye lo que se llama un

registro. Los registros se asocian formando ficheros que constituyen normalmente el objeto de los comandos del CP/M.

De hecho, cada orden admitida por el CP/M especifica un proceso particular al que deben someterse todos los bytes contenidos en los ficheros indicados por el usuario en la orden.

Como la dimensión de un registro lógico es de 128 bytes (equivalente a 2^7) y el número de registros que es posible alcanzar en un fichero es de 65.536 (2^{16}), la dimensión máxima de un fichero será 65.536 × 128, 8.388.608 (equivalente a 2^{23}).

El CP/M permite el acceso a los datos contenidos en un fichero en modo secuencial, o sea, uno tras otro, o en modo aleatorio (o directo), operando directamente sobre el registro que se haya especificado (con su número correspondiente).

La manera de identificar un fichero en ambiente CP/M se establece sintácticamente con una cadena de no más de ocho caracteres (letras mayúsculas del abecedario inglés o espacios blancos) que constituye su nombre, y otra cadena de no más de tres caracteres, que es la llamada "extensión". Ambas deben ir separadas por un punto ("."). Normalmente, al primer grupo de ocho caracteres (que indicaremos en general con "nombre-fichero") se le atribuye la tarea de especificar el nombre del fichero, y al segundo grupo (a la que nos referiremos como "ext"), el tipo de información contenida en el mismo.

Esta indicación se vuelve obligatoria para algunos tipos de ficheros, sobre los que sólo pueden llevarse a cabo funciones particulares si la extensión es una determinada.

Veamos algunos ejemplos de ficheros admitidos por el CP/M:

SUELDOS.NOV   PAGAS 11.84   NOVIEMBRE.984

En cambio no son válidos:

(SUELDO).NOV	Los paréntesis no son admitidos.
PAGASDENOV.1984	Nombre del fichero con más de 8 caracteres.
SUELDOS.NOVIEMBRE	Extensión con más de tres caracteres.
11=2?.NOV	El signo igual y la interrogación no son admitidos.

En caso de que se tengan que someter a una misma operación más de un fichero, podría resultar aburrido y no conveniente repetir muchas veces la misma orden, modificando sólo el fichero deseado. El CP/M, por ello, tiene una función particular, que podríamos denominar de enmascaramiento (o comodín, "joker").

Esta acción consiste en utilizar el asterisco o el signo de interrogación en el comando, con los siguientes fines:

a) Si se utiliza el signo de interrogación, éste sustituye a cualquier carácter situado en su misma posición; se pueden utilizar hasta 8 y hasta 3, en los campos del nombre y de la extensión, respectivamente.

Si, por ejemplo, sobre el disco están presentes los ficheros:

PAGAS1.NOV PAGAS2.NOV PAGAS11.NOV
PAGAS1.DIC BIGAS1.NOV

especificando:

??GAS?.NOV

el sistema hará que el comando correspondiente afecte a los ficheros:

PAGAS1.NOV PAGAS2.NOV BIGAS1.NOV

Los otros ficheros están excluidos; el cuarto, por tener una extensión diferente de la especificada, y el tercero, por tener un carácter más que el fichero especificado.

El CP/M, en el caso que un fichero tenga un nombre con menos de 8 caracteres, completa el grupo con espacios blancos (por lo que PAGAS?.NOV es lo mismo que PAGAS? .NOV).

b) Si, en cambio, se utiliza el asterisco, sustituye a cualquier grupo entero de caracteres, por lo que se comprenderán todos los ficheros en los que el nombre y la extensión presenten el mismo grupo de caracteres indicado en el fichero del comando antes o después del asterisco.

Utilizando el mismo ejemplo, si especificamos:

PAGAS*.NOV

equivale a decir:

PAGAS1.NOV PAGAS2.NOV PAGAS11.NOV

Como se puede notar, BIGAS1.NOV está excluido porque los dos primeros caracteres no son iguales a los indicados, y PAGAS.DIC también es ignorado porque no tiene la misma extensión.

Si indicamos entonces:

PAGAS*.*

nos referiremos a los primeros cuatro ficheros.

Naturalmente, serán totalmente equivalentes:

. y ????????.???

*.ext y ????????.ext

nombre-fichero.* y nombre-fichero.???

en donde se utilizan las letras minúsculas para especificar cualquier grupo de caracteres elegido por el usuario para ese campo.

La expresión:

.

es usada normalmente para especificar la totalidad de los ficheros presentes sobre el disco.

El CP/M es un S.O. utilizable por una sola unidad de comando *(consola)* pero con poder para conectarse a varias unidades de disco *(disk drive)*.

La conexión con las unidades de disco se obtiene a través de letras mayúsculas (de la A a la P) seguidas por los dos puntos, cada una asociada a una única unidad lógica.

En el momento de activarse, el CP/M muestra en la pantalla el "copyright" y la versión instalada, y se prepara a aceptar órdenes del usuario con la visualización de la letra correspondiente a la unidad que ha sido activada por el sistema (generalmente la A).

Normalmente el *prompt*, o sea, el símbolo que el sistema operativo visualiza en la pantalla para señalar al usuario que está preparado a aceptar sus órdenes, es un "mayor que" (>):

A>

Cada vez que una orden se refiera a un fichero y no se especifique la unidad de discos donde está, el CP/M lo buscará sobre el disco situado en la unidad activa (en el ejemplo, la "A").

Para activar otras unidades será necesario anteponer al nombre del fichero la letra correspondiente a la unidad deseada seguida de dos puntos. Si, por ejemplo, sobre el disco de la unidad B está almacenado el fichero CAMBIO.UNI, se necesitará el comando:

B:CAMBIO.UNI

para que el fichero sea hallado.

Si se quiere utilizar como unidad por defecto, (default, o sea,

como la unidad sobre la cual el S.O. obra si no hay explícita indicación) la unidad B, por ejemplo, se pulsará:

A>B:

y, seguidamente, la tecla RETURN (o ENTER). En la pantalla aparecerá:

B>

confirmando que el CP/M ha aceptado pasar a considerar la unidad B como activa.

Evidentemente, sólo es posible seleccionar las unidades existentes.

En el momento en que el usuario pulse caracteres en minúsculas se transformarán en sus equivalentes mayúsculas.

Escribiendo en el teclado podremos ver cómo en la pantalla se visualizan los caracteres pulsados y cómo no se lleva ninguna acción por el ordenador hasta que no se pulse la tecla RETURN. Sólo entonces el S.O. examina la corrección formal y la consistencia de lo que se ha escrito y lo lleva, en su caso, a cabo. A partir de ahora se sobreentenderá que al final de una secuencia de caracteres enviada por el usuario se pulse la tecla ENTER o RETURN, aunque al principio lo indicaremos algunas veces, para acostumbrarnos, con el símbolo ↵.

Comandos residentes y no residentes

La memoria central de un ordenador se puede comparar a un gran casillero. El acceso a una celdilla individual para depositar o coger información se obtiene simplemente con la indicación de su dirección.

Por suerte, no es deber del usuario el control y la optimación del uso de las celdas de memoria: el desagradable trabajo es desarrollado por entero por el microprocesador y el software de base.

En el curso de la ejecución de los programas normalmente utilizados, el usuario necesita preocuparse de cómo y por qué cosas está ocupada la memoria central. Además, puede ignorar que una parte de ella está reservada de manera exclusiva y no le es disponible.

Cuando hablamos antes de la activación (carga) del CP/M, queríamos referirnos a que el CP/M ocupa una parte de la memoria (y la vuelve reservada) para impedir que posteriores operaciones de grabación o escritura de datos borren el contenido e impidan la ejecución de las tareas "institucionales" del S.O.

El funcionamiento del ordenador está controlado por una serie de programas, o sea, por secuencias de instrucciones en lenguaje máquina. Sobre el disco, estos programas se presentan en forma de ficheros y, una vez cargados, se alojan en secciones de la memoria destinadas a este fin, sin que la integridad del sistema operativo corra peligro.

En el entorno CP/M existen dos tipos de comandos: residentes y no residentes.

- Los comandos residentes son parte integrante del CP/M y se hallan en la sección del S.O. inicialmente cargada en memoria.
- Los no residentes quedan sobre el disco como ficheros con la extensión COM (ej.: DDT.COM, DUMP.COM, PIP.COM, etc.) y, cuando son requeridos, se cargan en la zona llamada TPA (Transien Program Area), sección de la RAM reservada a los programas llevados a cabo bajo el control del CP/M.

Por tanto, los comandos no residentes son verdaderos programas que, para ejecutarlos, es suficiente especificar sólo el nombre (es decir, para ejecutar STAT.COM se pulsará sólo STAT).

Si, por lo tanto, no están presentes sobre el disco, es imposible su ejecución; además, la llamada a uno de ellos borra el que pudiera estar antes en la TPA.

En este libro haremos referencia a los comandos de la versión 2 del CP/M, la más difundida.

CAPITULO III

LOS COMANDOS RESIDENTES MÁS USADOS

DIR: el registro de los ficheros

La primera operación que normalmente se hace cuando nos encontramos ante un disco es intentar tener una idea del tipo de datos y programas que contiene. El comando más usado en el CP/M es, por tanto, el que permite visualizar en la pantalla la lista de todos los ficheros presentes en el disco. En inglés se denomina *directory* (DIR).

El usuario, para obtener en la pantalla el listado de todos los ficheros contenidos en el disco A, deberá pulsar:

A>DIR ↵ (el signo ↵ equivale al RETURN)

El formato en el que aparece la respuesta será, utilizando ficheros hipotéticos, como los ejemplos:

```
A: FILENUM1 TXT:FILENUM1 DOC: ROMA      160:TOKYO    J.64
A: MEXICO M68:MONACO D72:MONTREAL C76:MOSCU      A80
   A>
```

La información proporcionada sobre los ficheros contenidos en el disco A se limitan al nombre y a la extensión.

Otro uso común de DIR se tiene cuando queremos saber si un fichero cualquiera está contenido en el disco; en este caso a DIR le debe seguir el nombre del fichero buscado. Suponiendo que queramos saber si en el disco existe el fichero FILETEST.TXT, deberemos pulsar:

A>DIR FILETEXT.TXT

Si el fichero está presente, la respuesta del CP/M será:

A: FILETEST.TXT
A>

Si, en cambio, no está contenido en ese disco, en la pantalla se leerá:

NO FILE (es decir, no encuentro ese fichero).
A>

Es posible también utilizar la función de enmascaramiento. Así, supuesto que tenemos sobre el disco los mismos ficheros del capítulo 2, al dar:

A>DIR *.NOV ↵

se obtendrá la respuesta:

A: PAGAS1 NOV : PAGAS2 NOV : PAGAS11 NOV :
BIGAS1 NOV
A>

es decir, se listarán todos aquellos ficheros que contengan NOV como extensión, mientras que con

A>DIR PAGAS.* ↵

se obtendrá:

A: PAGAS1 NOV : PAGAS2 NOV : PAGAS1 DIC

El * hace, por tanto, de sustituto de cualquier serie de caracteres posibles. Así, las dos expresiones siguientes:

A>DIR ↵
A>DIR *.* ↵

son perfectamente equivalentes; visualizan en la pantalla la lista de todos los ficheros contenidos en el disco A.

DIR envía sus informaciones directamente a la pantalla. Para obtenerlas en la impresora hay que pulsar al mismo tiempo la tecla CONTROL y la tecla P (CTRL+P), para que, funcionando como interruptor, pase la información de la pantalla al papel.

Para obtener la lista de los ficheros contenidos en un disco distinto del actual de trabajo (en nuestro caso, por ejemplo, el de la unidad B) se puede ejecutar el comando

A>DIR B: ↵

o la secuencia:

A>B: ↵
B>DIR ↵

¿Hay alguna diferencia entre estas opciones? Sí: en el primer caso, permanecerá como unidad de disco de trabajo, por defecto, la A, mientras que en el segundo caso será la B, a menos que enviemos el comando

B>A: ↵

que restablece la situación precedente.

Igualmente, si lo que deseamos saber es si un determinado fichero existe sobre una unidad distinta de la corriente, podremos usar el nombre del fichero precedido de la especificación de la unidad. Por ejemplo, queriendo hallar la presencia del fichero MUESTRA.TXT sobre el drive B, se pulsará

A>DIR B:MUESTRA.TXT ↵

La unidad A quedará como la de defecto.

TYPE, una mirada al contenido

Los ficheros en disco pueden ser de dos tipos, muy distintos por características y funciones: ficheros compuestos por caracteres ASCII, que se pueden imprimir, y ficheros de código máquina (binario ó hexadecimal).

A la primera categoría pertenecen todos los ficheros generados por tratamiento de texto y, además, los que contienen la lista secuencial de las instrucciones que componen un programa, escritas en un lenguaje cualquiera (los llamados códigos-fuente).

Los programas escritos de esta manera sufren particulares transformaciones que los llevan a una forma más cercana a secuencias de instrucciones de código binario (para que la computadora actúe con más facilidad).

A la segunda categoría, la de los ficheros binarios, pertenecen todos los programas transformados a esta forma (son los códigos-objeto).

Esta distinción tiene un exacto sentido en el CP/M por cuanto, mientras los ficheros binarios pueden leerse y modificarse únicamente por medio del mismo proceso de ensamblaje que los ha generado, los de texto son fácilmente accesibles al usuario sin el empleo de instrumentos particulares de programación. Es el comando residente TYPE quien permite visualizar en la pantalla de la computadora el contenido del fichero indicado, línea tras línea.

Evidentemente, los datos de uno y otro tipo de ficheros están almacenados como una sucesión de números binarios ("0" y "1"); el nombre de binarios se les da también a los que contienen un código-objeto simplemente para indicar que en ellos esta sucesión no tiene ninguna correspondencia con cualquier otro código, mientras en los de código-fuente van asociados al código ASCII.

La sintaxis del comando es, tomando como muestra el fichero FILETEXT.TXT:

A>TYPE FILETEST.TXT ↵

la contestación del sistema operativo será inmediata, aunque válida sólo si el fichero existe en la unidad A.

Si el fichero es un código binario, en la pantalla aparecerán, en secuencias sin significado alguno y totalmente desordenados, los símbolos del set de caracteres de la máquina. Por otra parte, tampoco es de extrañar que el sistema se bloquee en este caso, obligando a apagar y volver a conectar el ordenador.

Por contra, un fichero de caracteres ASCII será visualizado exactamente tal y como resulta escrito en el disco.

En el caso de un fichero de longitud normal es probable que no entre en su totalidad en la pantalla: el S.O. no se interesa lo más mínimo por este problema y seguirá visualizando línea tras línea el fichero, provocando la desaparición de los caracteres que ocupaban la posición extrema en la parte alta de la pantalla para dejar sitio a las nuevas líneas, que desplazan hacia arriba las anteriores. En principio esto haría ilegible todo el fichero excepto las últimas líneas (aquellas que quedarían en la pantalla al final). Para evitar este inconveniente y poder leer tranquilamente el contenido del fichero se acostumbra a detener la visualización del fichero pulsando al mismo tiempo las teclas CONTROL y S (CTRL+S); inmediatamente la sucesión de líneas en la pantalla se interrumpirá. Para que continúe basta pulsar una tecla cualquiera; así se irán sucediendo pantallas una tras otra.

También se puede explotar con TYPE la función de enmascaramiento para conseguir la visualización secuencial de todos

los ficheros correspondientes a la "mascarilla" utilizada. Así, la orden:

A>TYPE * * ↵

servirá para observar en pantalla el contenido de todos los ficheros del disco.

El comando TYPE, por las escasas posibilidades de control que ofrece, se utiliza sólo para breves consultas de los ficheros (para ver cómo hacíamos esa entrada, cuál era ese dato que no recordábamos...). Para tareas de mayor complejidad se prefieren a él otros comandos, así como para la impresión (ver PIP y XSUB).

ERA, borrado de ficheros.

El control completo de un disco comprende la posibilidad de borrar informaciones que ya no interesan (*erase* significa borrar en inglés).

El comando tiene efecto inmediato; por lo tanto, después de utilizarlo sobre un fichero será imposible hallar o intentar leer de nuevo el mismo: habrá desaparecido del directorio.

Probablemente es el comando en el que se utiliza mejor la función de enmascaramiento, lo que permite un gran ahorro de tiempo (aunque también conlleva sus peligros). De hecho, siguiendo con el ejemplo de los ficheros del capítulo 2, si se envía la orden:

A>ERA PAGAS?.NOV ↵

al hacer DIR, inmediatamente después el sistema responderá:

```
A>DIR ↵
A: PAGAS11   NOV : PAGAS1    DIC : BIGAS 1    NOV
A>
```

es decir: PAGAS1.NOV y PAGAS2.NOV han sido borradas.

En este comando, equivocarse al especificar un fichero puede provocar efectos desastrosos. Si el error es "enorme" se generarán las oportunas señalizaciones por parte del sistema operativo. Así, en el caso que se envíe la fatal orden

A>ERA *.*

el sistema operativo exige la confirmación, emitiendo la pregunta:

Are you sure? (Y/N)   (es decir, ¿Estás seguro? (S/N)).

Es la última oportunidad para el usuario distraído que ha olvidado que una orden de este tipo destruye todos los ficheros presentes en el disco.

Naturalmente, en caso de que la confirmación se lleve a cabo —pulsando la tecla Y (o sea, Yes)—, el CP/M ejecuta tranquilamente la orden; pulsando la tecla N (o sea, No) o cualquier otra que no sea Y se anula la orden.

Si en un disco han sido destruidos todos los ficheros, la respuesta a una petición posterior del directorio será:

```
A>DIR ↵
NO FILE        (es decir: ningún fichero presente).
A>
```

Observación importante.—Dado el singular mecanismo con el que el S.O. graba el contenido de un fichero, el borrado consiste sólo en la eliminación de su nombre del directorio, y no en el borrado físico real del espacio ocupado por el fichero. Dicho con más sencillez: los datos se quedan en principio en el disco tal y como estaban escritos, pero el S.O. no los deja más a disposición del usuario, y al declarar no reservada la zona ocupada por ellos, confía a sucesivas grabaciones el deber de borrarlos definitivamente.

Esto explica la rapidez con la que siempre se efectúa la función de cancelación, así como la posibilidad (desarrollada con la ayuda de complejos programas) de volver a "recomponer" el fichero siempre que no se realice ninguna operación de grabación tras borrarlo.

Por evidentes motivos de seguridad, el comando ERA no puede destruir el núcleo del S.O. ni los comandos residentes.

REN: dando nuevo nombre a los ficheros

Otra función extremadamente útil en el control de los ficheros es la que permite cambiar el nombre de un fichero (en inglés *rename* equivale a dar un nombre nuevo).

El comando REN permite la modificación del nombre o la extensión de un fichero o de ambas cosas a la vez, sin alterar lo más mínimo su contenido o su posición en el disco.

Naturalmente, para este comando no es válida la función de enmascaramiento, que sólo crearía confusión.

Si, por ejemplo, se quiere modificar el nombre del fichero FILETEST.TXT y pasarlo a MUESTRA.DOC, deberemos hacer lo siguiente:

```
A>REN MUESTRA.DOC=FILETEST.TXT ↵
```


El CP/M se encarga de controlar si existe ya un fichero con el nombre que se quiere asignar, en cuyo caso visualiza el mensaje:

```
FILE EXISTS        (Ya existe un fichero con ese nombre.)
A>
```

Tratándose de modificación, y no de traslado, el comando no hace distinción entre el disco sobre el que está el fichero viejo y el disco sobre el que se ha de crear el fichero nuevo. Por lo tanto, en caso de que no sea indicada ninguna unidad distinta, el fichero viejo es buscado sobre la unidad actual y el nuevo será creado en ella.

En caso de querer trabajar con distinta unidad, será suficiente con indicarla antes que cualquiera de las dos especificaciones. Son, por tanto, perfectamente equivalentes, en caso de la unidad B:

```
A>REN B:MUESTRA.DOC=FILETEST.TXT ↵
A>REN MUESTRA.DOC=B:FILETEST.TXT ↵
A>REN B:MUESTRA.DOC=B:FILETEST.TXT ↵
```

Una orden con dos unidades distintas como

```
A>REN A:MUESTRA.DOC=B:FILETEST.TXT ↵
```

no será admitida.

CAPITULO IV

LOS COMANDOS NO RESIDENTES MÁS ÚTILES

STAT: un poco de todo

El CP/M ha nacido para solucionar, entre otras cosas, los problemas de conexión del ordenador con sus periféricos. Por lo tanto es importante que el usuario conozca y pueda modificar de acuerdo con sus necesidades las modalidades con las que son realizados estos enlaces.

El comando STAT es el más utilizado entre los comandos no residentes (o sea, presentes sobre un disco como un fichero, con la extensión COM).

Proporciona informaciones y permite la modificación de los parámetros con los que se realizan los enlaces lógicos con los periféricos (naturalmente, los enlaces físicos incumben exclusivamente al *hardware*).

Si tecleamos el comando STAT sin ninguna otra especificación, el CP/M visualizará algunas informaciones de tipo general sobre los discos presentes en las unidades enlazadas:

```
A>STAT↵
```

operando sobre un sistema compuesto de dos unidades de disco *(disk drive)* provocará una respuesta del tipo:

```
A:R/O,SPACE:120K
B:R/W,SPACE:80K
A>
```

en donde la expresión R/O *(Read/Only,* sólo lectura) indica que el CP/M permite al usuario exclusivamente leer los datos conte-

nidos en el disco A, pero que protege toda la superficie del disco contra cualquier intento de escribir sobre él. Así, pues, el S.O. impide cualquier alteración del estado magnético del disco. La protección es eficaz, ya que impide escribir, borrar y modificar datos.

La expresión R/W *(Read/Write,* lectura y escritura), por contra, indica que es posible efectuar sobre el disco situado en la unidad indicada cualquier tipo de operación. El término SPACE seguido de un número y la letra K indica al usuario la cantidad de espacio, en kilobytes, que resta a su disposición sobre el disco. En el ejemplo anterior, por lo tanto, el usuario podría utilizar todavía 120 × 1.024 bytes (120 Kbytes) en el disco de la unidad A, mientras que sobre el disco de la B el espacio libre corresponde a 80 × 1.024 bytes.

Si estamos interesados en conocer únicamente el espacio libre sobre un disco, por ejemplo el contenido en la unidad B, deberemos pulsar:

A>STAT B: ←

para obtener:

BYTES REMAINING ON B: 104K (104 Kbytes disponibles en B:)
A>

En este momento es necesario realizar una aclaración sobre el mecanismo en base al cual el CP/M establece si un disco tiene o no que ser protegido. Como para acceder a los datos de un disco es suficiente especificar la unidad en la que él va a operar, podría parecer posible la sustitución de un disco en el transcurso de una sesión de escritura, pero no es así. El CP/M en el momento de enlazarse con una unidad de discos tras una inicialización, copia en una sección de la memoria central el directorio del disco presente. En todos los sucesivos requerimientos de acceso al disco el CP/M verifica que el directorio del disco sea el mismo que el anteriormente grabado por él.

En caso de disparidad, deduce que el disco ha sido sustituido y, para evitar problemas, bloquea cualquier operación posterior de escritura que le sea ordenada, marcando con R/O el disco en cuestión.

Para restablecer el estado inicial y, por lo tanto, permitir la escritura a través de esa unidad, es necesario que el CP/M borre el directorio grabado anteriormente y lo sustituya por el nuevo.

Para hacer partir el CP/M de ese primer estado hay que pulsar al mismo tiempo las teclas de CONTROL y C (CTRL+C). Esta acción, conocida como *warm bott,* o inicialización en caliente, vuelve a marcar al disco con R/W y consiente la variación de los datos situados en su superficie.

Es obvio que este mecanismo puede ser también aprovechado para proteger discos que contengan datos de relevante importancia. Si, por ejemplo, se quiere impedir la modificación accidental de los datos contenidos en el disco de la unidad A, se pulsará:

A>STAT A:=R/O

y la protección mantendrá sus efectos hasta la siguiente inicialización (CTRL+C o, como se suele escribir, ^C).

Un uso muy frecuente del comando STAT es en la obtención de informaciones adicionales sobre los ficheros. En efecto, el comando DIR no proporciona, de hecho, ninguna información relativa a la longitud o características de los ficheros.

Hagamos una prueba con la función de enmascaramiento:

A>STAT A:*.*

La respuesta proporcionará toda la información relativa a los ficheros del disco de la unidad A. Utilizando algunos ficheros del capítulo 2 como ejemplo, en la pantalla se tendrá:

Recs	Bytes	Ext	Acc
32	4K	1	R/W A:PAGAS1.DIC
48	6K	1	R/W A:PAGAS1.NOV
64	8K	1	R/W A:PAGAS11.NOV
6	1K	1	R/W A:PAGAS2.NOV

A>

con los ficheros en orden alfabético.

Deberemos utilizar esta función también cuando queramos simplemente leer el directorio con los ficheros en orden alfabético (o imprimirlo tras haber pulsado CTRL+P).

En el campo Recs (Records, registros) aparece el número de registros de 128 bytes que ocupa el fichero.

En el campo Bytes aparece el número de kilobytes del fichero (puede verificar que Bytes=Recs*128/1.024).

En el campo Ext aparece el número de bloques, de 16K, ocupados por el fichero (Ext=Bytes/16 aproximando por exceso).

En el campo Acc aparece la modalidad de acceso del fichero además del conjunto de caracteres utilizados como identificadores del nombre y extensión.

Ahora bien, las características relativas al modo de acceso son modificables con el mando STAT. Así podemos hacer:

A>STAT A:PAGAS1.DIC $DIR
A:PAGAS1.DIC SET TO DIR

A>STAT A:PAGAS1.NOV $SYS
A:PAGAS1.NOV SET TO SYS

A>STAT A:PAGAS11.NOV $R/O
A:PAGAS11.NOV SET TO R/O

A>STAT A:PAGAS2.NOV $R/W
A:PAGAS2.NOV SET TO R/W

A>

La primera forma establece que el fichero (o los ficheros) objeto de comando se visualizarán durante la ejecución del comando DIR.

La segunda determina lo contrario: el fichero no se visualizará en el curso de la ejecución de un comando DIR.

La tercera permite exclusivamente la lectura del fichero, pero no la escritura sobre él.

La cuarta forma fija el atributo de lectura/escritura del fichero.

Las características de los ficheros que hemos mencionado en nuestro ejemplo se visualizarán así:

A>STAT A:*.*

Recs	Bytes	Ext	Acc
32	4K	1	R/W A:PAGAS1.DIC
48	6K	1	R/W (A:PAGAS1.NOV)
64	8K	1	R/O A:PAGAS11.NOV
6	1K	1	R/W A:PAGAS2.NOV

A>

En caso de que se especifique el comando:

A>STAT A:*.* $S

el formato de la respuesta será:

Size	Recs	Bytes	Ext	Acc
65536	32	4K	1	R/W A:PAGAS1.DIC
48	48	6K	1	R/W (A:PAGAS1.NOV)
64	64	8K	1	R/O A:PAGAS11.NOV
6	6	1K	1	R/W A:PAGAS2.NOV

El dato correspondiente a Size (tamaño) especifica la longitud virtual del fichero en número de registros; se dice virtual porque el fichero puede no tener ocupado todo ese espacio en realidad. En el caso de ficheros de acceso secuencial, Size y Recs son iguales, mientras en el caso de ficheros de acceso aleatorio (o directo), Recs es el número de registros efectivamente ocupados y Size indica el máximo número de registros que se pueden contener en el fichero; esto ocurre porque al no existir el vínculo de la secuencialidad entre los distintos registros, es posible que entre uno y otro se encuentren algunos vacíos.

En el ejemplo se ha considerado PAGAS1.DIC como un fichero de acceso aleatorio. En el momento de la creación de un fichero los atributos por defecto son DIR y R/W.

Con STAT es posible tener una idea general de las características de la unidad controlada. Si enviamos

A>STAT A:DSK:

la contestación será del tipo:

A:	Drive Characteristics
65536:	128 Byte Record Capacity
8192:	Kilobyte Drive Capacity
128:	32 Byte Directory Entries
0:	Checked Directory Entries
1024:	Records/Extent
128:	Records/Block
58:	Sectors/Track
2:	Reserved Tracks

A>

Así el usuario sabrá que la unidad enlazada es la A, su capacidad es 8.192 kilobytes (65.536 × 128 bytes), 128 el número de ficheros admitidos y 0 el número de ficheros existentes, 1.024 registros por cada Extent, 128 por cada bloque, 58 sectores (sectors) por cada pista (track) del disco y dos pistas reservadas al S.O. y no accesibles al usuario.

Los valores que tomen estas informaciones dependen totalmente de la versión del CP/M instalada y de los periféricos enlazados.

El comando STAT ofrece la posibilidad de variar la manera en que un dispositivo se enlaza a un ordenador. Por lo que se refiere a los periféricos, el CP/M opera considerando que existen cuatro dispositivos enlazados de manera lógica:

CON: consola, interfaz entre usuario y CP/M
LST: dispositivo de impresión
PUN: dispositivo de perforación
RDR: dispositivo de lectura de banda perforada

El movimiento de los datos de entrada y salida se materializa en los correspondientes dispositivos físicos:

TTY: dispositivo tipo telex (de baja velocidad)
CRT: unidad vídeo
LPT: impresora
PTR: lector de cinta de papel (de alta velocidad)
PTP: perforador de cinta de papel (de alta velocidad)
BAT: unidad secuencial de entrada
UC1: disponible por el usuario
UL1: disponible por el usuario
UR1: disponible por el usuario
UR2: disponible por el usuario
UP1: disponible por el usuario
UP2: disponible por el usuario

No todos los dispositivos lógicos pueden ser asignados a cualquier dispositivo físico. Para conocer las correspondencias permitidas se debe pulsar:

A>STAT VAL:

orden que proporciona un cuadro general de las posibles asignaciones:

```
Temp R/O Disk d:$ R/O
Set indicatorfilename.typ $ R/O $ R/W $ SYS $ DIR
Disk Status: DSK: d:DSK
Iobyte ASSIGN:

CON = TTY: CRT: BAT: UCI:
RDR = TTY: PTR: UR1: UR2:
PUN = TTY: PTP: UP1: UP2:
LST = TTY: CRT: LPT: UL1:

A>
```

Las tres primeras líneas recuerdan las posibilidades ofrecidas por el comando STAT, ya discutidas anteriormente, mientras que Iobyte Assign presenta el conjunto de los posibles emparejamientos para las unidades de I/O (Input/Output).

Para saber cuáles son las asociaciones establecidas se envía la orden:

A>STAT DEV:

Un ejemplo de contestación será:

CON: =CRT: (la pantalla es normalmente la unidad de salida)
RDR: =UL1:
PUN: =TTY:
LST: =LPT: (la impresora es otra unidad de salida típica).

Las asignaciones lógico-físicas pueden ser modificadas en cualquier momento. Por ejemplo, se puede hacer más rápida la puesta a punto de algunos programas de impresión sacando por pantalla todos los datos que deberían imprimirse sobre el papel. Con:

A>STAT LST:=CRT:

el sistema verá entonces la unidad de visualización como el dispositivo de impresión y enviará sobre pantalla todos los caracteres a imprimir.

PIP, el genio de los "traslados"

El CP/M está dotado de un potente y flexible instrumento para el traslado de los ficheros a los periféricos: el comando PIP (Peripheral Interchange Program).

Es posible aprovechar las opciones ofrecidas por el comando de dos maneras distintas:

a) Para disminuir el tiempo y tener que pulsar menos teclas cuando se tienen que efectuar distintas secuencias de traslado, se puede enviar:

A>PIP

La respuesta será:

*_

El asterisco es una marca (prompt) del comando PIP, que permite escribir todas las líneas que queramos del comando sin tener que repetir cada vez PIP; se permanece en el ámbito del programa de traslado hasta que se pulsa la tecla

RETURN sin ser precedida de ninguna otra especificación. Esta última acción hace finalizar la sesión de traslado y hace aparecer de nuevo:

A>

b) Tecleando PIP seguido de la línea de comando deseada, la ejecución se llevará a cabo inmediatamente, pero se volverá al CP/M una vez acabada.

La forma general del comando PIP es:

destino=origen1,origen2,...,origenN

en donde *destino* puede ser el nombre nuevo de un fichero o dispositivo, es decir, el lugar donde acabarán los datos a trasladar, mientras la secuencia de origen1, origen2, etc., es el grupo de ficheros o dispositivos que representa la fuente de la que se sacan estos datos.

Consideremos que queremos crear un duplicado del fichero PAGAS1.NOV con el nombre de COPIA2, y que tenemos sobre el disco de la unidad A el programa PIP.COM. La forma de proceder sería:

```
A>PIP
*COPIA2=PAGAS1.NOV
*_
```

Puede comprobar que la ejecución de esta orden no destruye el fichero de origen, sino que crea una copia con distinto nombre. Si quiere crear sobre el disco B el duplicado de un fichero que está en el disco de la unidad A, debería poner:

*B:COPIA2=PAGAS1.NOV

En efecto, si falta la indicación de la unidad, se entiende que el fichero debe ser buscado sobre la de defecto (en el ejemplo, la A).

Si el fichero a trasladar queremos que conserve el mismo nombre sobre el disco de destino, es suficiente con especificar tan sólo la unidad sobre la cual se efectuará la copia. Por ejemplo, después de:

*B:=A:PAGAS1.NOV.

existirán sobre los discos A y B copias idénticas, por contenido y nombre, de PAGAS1.NOV.

Si, en cambio, es necesario crear sobre el disco de la unidad B un fichero que sea la concatenación de ficheros presentes en unidades distintas, se puede hacer:

*B:TOTAL=B:COPIA2.NOV,A:PAGAS2.NOV

con lo que se obtiene la creación del fichero TOTAL constituido por el contenido del fichero COPIA2.NOV presente sobre el disco de la unidad B y por el del fichero PAGAS2.NOV presente sobre el disco situado en la unidad A. Muchas veces querremos copiar sobre un disco todos los ficheros presentes sobre otro. El camino más rápido posible es utilizar la máscara *.*, por ejemplo:

B:=A:.*

carga en la unidad B todos los ficheros contenidos sobre el disco de la unidad A.

Escribir:

A>PIP B:=A:PIP.COM

es una secuencia perfectamente lícita, ya que PIP.COM es un comando no residente del CP/M y, por lo tanto, es, a todos los efectos, un fichero como los demás.

La acción de traspaso de ficheros no es permitida, sin embargo, con respecto al CP/M, el cual, por decisión soberana, va a resultar no accesible a sus vasallos, los comunes programas.

Otro útil empleo de PIP es el trasladar ficheros de un dispositivo, físico o lógico, que esté en situación de generar datos (están excluidos, por tanto, los que tienen únicamente salida como LST: o RDR:) a otro dispositivo que esté en situación de aceptarlos (excluye, por tanto, a aquellos asociados a PUN:).

De hecho, si quiere imprimir un fichero basta pulsar

*LST:=PAGAS1.NOV

Obtendrá sobre el papel la lista de los datos contenidos en el disco presente en la unidad A, bajo el nombre de PAGAS1.NOV.

Si, en cambio, envía la orden

*MANDOS=CON:

el efecto será mandar a un fichero llamado MANDOS todo lo que el usuario escriba en la consola. Este funcionamiento desaparece en cuanto se pulsan al mismo tiempo las teclas CONTROL y Z, acción que provoca la señal de fin de fichero (EOF=end of file).

Lo práctico de esta solución nos permite redactar con gran sencillez simples ficheros de texto sin tener que recurrir a complejos programas de vídeo-escritura. Un ejemplo aclarará el procedimiento. Creemos un fichero DEMUESTRA.TXT que contenga el texto

Los Miserables
Los misterios de París

Representando la pulsación de la tecla de CONTROL con el símbolo ^, el procedimiento operativo que requiere es:

```
A>PIP DEMUESTRA.TXT=CON:
Los miserables^M^J
Los misterios de París^M^J
^Z
A>
```

Para fines y funciones particulares son admitidos otros nombres de dispositivos, entre los que es útil recordar PRN.

El uso de este dispositivo es, de hecho, especialmente apto para imprimir los listados de programas-fuentes, ya que sustituye por 8 espacios el carácter de tabulación, numera las líneas y las distribuye en páginas compuestas cada una por 60 líneas. Su empleo es la manera razonable de imprimir un fichero en papel, en lugar del verdaderamente poco aconsejable método de pulsar ^P (que activa el "eco" sobre la impresora —todo lo que introduzcamos por teclado saldrá por impresora—) seguido de TYPE.

Los otros dispositivos especiales son:

a) NUL: utilizado para enviar 40 caracteres NUL;
b) EOF: utilizado para forzar el fin de fichero sobre un dispositivo;
c) INP: dispositivo especial modificable por el usuario;
d) OUT: dispositivo especial modificable por el usuario.

La línea de comando de PIP puede ser completada por distintas opciones comprendidas entre corchetes.

Es posible especificarlas mediante letras minúsculas o mayúsculas, ya que producen los mismos efectos.

La opción más utilizada y que deberemos "optar obligatoriamente" en caso de copia de ficheros sobre disco (aunque ya se sabe que los buenos consejos están hechos para no seguirlos) es la que permite verificar si la escritura sobre el disco ha ido bien. Por ejemplo:

```
A>PIP B:=*.* [V]
```


lleva a cabo la copia de los ficheros de A sobre B, con un poco más de lentitud, pero asegura que la copia ha sido hecha correctamente.

Cuando se quiera copiar sólo la primera parte de un fichero y se conozca la cadena con la que se desea que el fichero acabe (por ejemplo, queremos que "esto es todo" sea la última cadena de un fichero de texto llamado MEMORIE.DOC), podemos hacer:

```
A>PIP
*A:=B:MEMORIE.DOC [Qesto es todo.^Z]
```

a continuación de lo cual el fichero MEMORIE.DOC del disco A será copia del contenido en el disco B excepto por los caracteres siguientes a la cadena indicada.

Si queremos lo contrario, conservar sólo la última parte de un fichero, se especifica la cadena a partir de la cual debe comenzar el nuevo fichero:

```
A>PIP
*A:=B:MEMORIE.DOC [Sesto es todo.^Z]
```

Una correcta ejecución de la orden requiere necesariamente seguir la secuencia mostrada.

La forma alternativa (PIP d:=p:nomfiche[Qcadena^Z]) traduce a letras mayúsculas todas las letras minúsculas que pertenecen a la cadena especificada y hacen posible el hallazgo.

La opción E permite leer sobre la pantalla el contenido del fichero objeto de traslado.

```
A>PIP A:=B:*.*[VE]
```

permite al usuario verificar que se ha llevado a cabo correctamente la copia y visualizar simultáneamente en pantalla la naturaleza de los datos trasladados.

La opción Pn divide, durante la impresión, el fichero en páginas de "n" líneas (60 por defecto). Admitiendo que queremos imprimir el fichero MEMORIE.DOC en páginas de 48 líneas, se enviará:

```
A>PIP LST:=B:MEMORIE.DOC(P48)
```

La opción N introduce el número de secuencia antes de cada línea y la opción Tn asocia a cada tabulación su correspondiente secuencia de "n" caracteres. Prácticamente se pueden considerar equivalentes las dos expresiones siguientes:

A>PIP LST:=B:MEMORIE.DOC(T8NP)
A>PIP PRN:=B:MEMORIE.DOC

Las opciones U y L transforman, respectivamente, todas las letras minúsculas en mayúsculas y todas las mayúsculas en minúsculas durante el traslado. Un uso particular está reservado a la opción O, que permite trasladar los ficheros que no estén compuestos por caracteres ASCII imprimibles. Recordemos que un fichero ASCII (de caracteres de texto) termina con el carácter ^Z, a diferencia de un fichero binario. Efectivamente, en la operación de concatenación de un fichero el CP/M trabaja simplemente removiendo los caracteres de ^Z situados al final de los ficheros intermedios y dejándolo exclusivamente al final del fichero resultante. Si bien es cierto que esto va bien con los ficheros de texto, hace imposible un uso correcto de esta función con los ficheros binarios. La opción O permite al comando PIP enlazar ficheros binarios sin problemas.

Si queremos crear un fichero binario TOTAL.ASM como suma de PARCIAL.1 y PARCIAL.2, ambos binarios, se deberá ejecutar:

A>PIP TOTAL.ASM=PARCIAL.1,PARCIAL.2(O)

para obtener la correcta finalización de la orden.

Otra opción importante es la que permite trasladar los ficheros que han sido creados en formato hexadecimal y que tienen como extensión HEX (es el mismo CP/M el que, mediante una función particular (ASM) permite la creación de estos ficheros).

La opción H ejerce controles particulares sobre los ficheros compuestos de caracteres hexadecimales, volviéndose prácticamente insustituible para este fin. Otras opciones permiten ignorar las características de accesibilidad de los ficheros (o sea R/O y SYS): la opción W permite escribir sobre un fichero designado como Read/Only y la opción R permite trasladar ficheros que no aparecen cuando se ejecuta el comando DIR.

Menos utilizadas son las opciones:

- B realiza el traslado de los datos en bloque;
- Dn permite, cortando todas las columnas más allá de la enésima, enviar datos a un dispositivo que controle sólo "n" columnas;
- F elimina los avances automáticos de página;
- I ignora todos los caracteres nulos de un fichero.

Después de haber examinado las opciones de PIP, alguien podría deducir que el comando TYPE es inútil o casi. De hecho, obliga a leer todo el fichero desde el principio, mientras PIP permite imprimir sólo el número de líneas deseadas.

Cojamos como ejemplo el fichero de texto MUESTRA.TXT y pulsemos:

A>PIP TRANSITO.TXT=MUESTRA.TXT[N]

de esta manera habremos creado un fichero en el cual todas las líneas están precedidas por su número de orden.

Pasamos ahora a decidir los parámetros de impresión, es decir, el número de líneas por página y las columnas de tabulación:

A>PIP LST:=TRANSITO.TXT[Tn1Pn2Sn3^ZQn4^Z]

que es, sin duda, más complejo que TYPE, pero permite seleccionar:

n1 como número de columnas del carácter de tabulación;
n2 como el número de líneas que forman cada página;
n3 como número de línea por donde ha de empezar la impresión;
n4 como número de línea con el que acaba la impresión.

Además, sustituyendo CON: por LST: se obtendrán los resultados sobre la pantalla y no en la impresora.

Finalmente, podemos decidir destruir TRANSITO.TXT o conservarlo para sucesivas impresiones.

SUBMIT, comandos a medida

A veces, es útil o necesario utilizar una secuencia de comandos del CP/M como si fueran instrucciones de un programa. Por ejemplo, si una secuencia es usada frecuentemente por un operador, sería conveniente dar un nombre a la misma y ejecutarlo como si fuera un simple comando. En el CP/M esto se logra con el comando SUBMIT, que nos permite crear un comando nuevo formado en base a otros del CP/M.

Para ello se crea un fichero que contenga línea tras línea los comandos que deseamos utilizar, en la misma secuencia en la que el usuario debería ejecutarlos. Este fichero tiene que ser creado con la extensión .SUB y contener las órdenes en forma sintácticamente correcta.

El comando SUBMIT permite al ordenador identificar los comandos del fichero creado, sin que el usuario tenga que escribirlos cada vez en el teclado.

El formato del comando es:

SUBMIT nombre-de-fichero V1, V2, V3...

donde:

nombre-de-fichero → nombre de un fichero de texto de líneas de comandos, cuya extensión es .SUB

V1, V2, V3, valores opcionales para ser sustituidos en las variables del fichero.

Puede suceder que siempre tengamos la necesidad, tras haber encendido el ordenador, de habilitar una impresora conectada a la salida paralela del ordenador, leer el directorio de la unidad B y ejecutar el programa MUESTRA.COM. Utilizaremos entonces el SUBMIT como sigue: En primer lugar se crea (mediante un programa de edición de textos) un fichero, al que vamos a asignar el nombre de INICIO.SUB, que contenga:

```
STAT LST:=LPT:
DIR B:
MUESTRA
```

Y ya está. Para ejecutar esta secuencia de comandos basta escribir:

```
A>SUBMIT INICIO
```

(nótese que la extensión SUB es superflua a la hora de la ejecución), con lo cual todo se desarrolla exactamente como si el usuario enviara una tras otra las órdenes contenidas en el fichero INICIO.SUB.

SUBMIT, nada más pedírselo transforma el fichero .SUB en un fichero temporal SSS.SUB del que obtiene las órdenes y que es borrado al final de la ejecución o cuando se produce una situación de error.

De todas maneras debemos puntualizar que el fichero con extensión .SUB tiene que estar presente sobre el disco en la unidad A para que todo discurra correctamente.

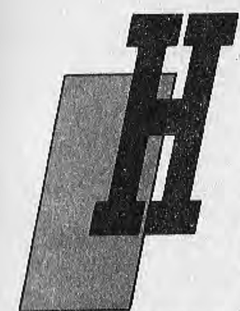
La agrupación de más de un SUBMIT es perfectamente realizable: basta con insertar en una línea del fichero .SUB otro comando SUBMIT.

Una cuidadosa utilización del SUBMIT permite definir nuevas funciones compuestas (las llamadas "macros") útiles en determinados contextos. Volveremos sobre este tema.

CAPITULO V

FUNCIONES AUXILIARES

DUMP, la verdadera cara de los ficheros



Hemos visto ya que es posible visualizar e imprimir los ficheros de texto de manera comprensible, mientras que no es factible someter a estas operaciones a los ficheros binarios.

En efecto, los ficheros binarios están escritos en la forma más apta para el ordenador: en código máquina; contienen caracteres que, en lugar de corresponder a una letra o a un número, van asociados a una particular función en el hardware de la CPU y pueden, por tanto, modificar el estado del ordenador al ser ejecutados.

El comando no residente DUMP visualiza en la pantalla el contenido de cualquier tipo de fichero, transformando cada carácter en el número que le ha sido asignado en el código correspondiente ASCII aceptado por el ordenador.

Los ficheros son expuestos en líneas de dieciséis caracteres cada una.

Los números de código visualizados no están expresados en la numeración decimal, sino en la hexadecimal, más conforme a la numeración en base 2 del ordenador.

Al final de cada línea aparecen los 16 símbolos que corresponden a los caracteres visualizados, con la particularidad que todos los caracteres que corresponden a símbolos no imprimibles son sustituidos por un punto.

Para llevar a cabo este comando debe darse el nombre completo del fichero, incluso su extensión. Supongamos que queremos trabajar con el fichero MUESTRA.TXT, creado con un programa de control de textos, cuyo contenido es:

Ejemplo de texto muestra.
La secuencia será:

A>DUMP MUESTRA.TXT

que hará aparecer en pantalla:

```
0100  45 6A 65 6D 70 6C 6F 20 64 65 20 74 65 78 74 6F   Ejemplo
0110  6D 75 65 73 74 72 61 2E 0D 0A 1A 00 00 00 00 00   de texto
                                                         muestra.
```

ofreciéndonos estas informaciones:

- el fichero, para ser procesado, se carga en la memoria central a partir de la posición 100 (hexadecimal);
- en la primera posición (100) encontramos la letra "E", en la 101 la "j", en la 102 la "e", etc.;
- a la letra E corresponde el número 45 (hexadecimal) en el código ASCII, a la "m" el 6D, etc.;
- el fichero termina con tres caracteres de control: ØD (línea nueva), ØA (continuación de línea) y 1A (final del fichero).

Cuando se trabaja con ficheros binarios ya no existe una correspondencia directa entre las combinaciones de símbolos del lenguaje máquina y las de las estructuras lingüísticas usuales, es decir: no se obtienen palabras con sentido en binario o en hexadecimal y la visualización de los caracteres a la derecha de la línea no tiene ninguna utilidad.

Para DUMP se puede repetir lo dicho para TYPE: mejor que él pueden actuar otros comandos (por ejemplo, DDT), aunque su empleo sea más complejo.

SAVE, guardando ficheros

El comando residente SAVE tiene una sintaxis muy particular:

SAVE n nombre-de-fichero.ext

De hecho, el ejemplo

A>SAVE 5 NEWFILE.TXT

obliga al CP/M, en este caso específico, a crear sobre el disco de la unidad A un fichero llamado NEWFILE.TXT que recoja todos los

caracteres contenidos en las primeras 5 (decimal) páginas de la memoria central, empezando por la posición 100 (hexadecimal).

Como cada página de memoria está compuesta por 256 bytes, la expresión utilizada en el ejemplo equivale a indicar que todos los caracteres comprendidos en la memoria entre la posición 100H (se indica con H la base hexadecimal) y la 5FFH constituirán el fichero especificado.

El uso de este comando es frecuente cuando se quiere mantener una disposición de la memoria para recuperarla posteriormente o para registrar las modificaciones llevadas a cabo en un programa.

Normalmente, más que la dimensión efectiva del fichero, se conoce la dirección final entre todas las ocupadas por el fichero después de haber sido cargado en memoria. Considerando que se debería restar 100H a esta dirección y luego añadir todas las posiciones necesarias para completar una página de memoria y saber así cuántas necesitamos, es más conveniente utilizar un método simplificado para calcular el número a especificar en el comando. Se empieza considerando que 100H es un múltiplo de 10H (base de la numeración hexadecimal) en cuanto equivale a 10H elevado a la segunda potencia. Esta consideración permite evitar la división entre números en base hexadecimal. Así, en el número hexadecimal correspondiente a la dirección se pueden ignorar tranquilamente las últimas dos cifras de la derecha. Esto equivale a dividir por 10H a la segunda potencia, ya que la lógica de las cuatro operaciones es independiente de la base de numeración adoptada; dividir por la base a la segunda potencia significa mover la coma dos cifras a la izquierda. Veamos un ejemplo. Supongamos que tenemos un fichero que ocupa CF9A posiciones. Descartando las dos últimas cifras de la derecha, se debe sólo transformar en decimal el número CF.

CF equivale a la expresión:

$C*10H^1+F*10H^0$

F corresponde a la cifra de la unidad, por lo tanto F (que en decimal vale 15) se multiplica por 1 ($10H^0=1H$).

C corresponde a la cifra de las dieciséis unidades, es decir, a la base de numeración; por lo tanto, C (que en decimal vale 12) se multiplica por 16 ($10H^1=10H$).

El resultado será:

$F=15*16^0=15*1=15$
$C=12*16^1=12*16=192$

y por tanto, CF=192+15=207.

Este es el número a especificar

A>SAVE 207 NEWFILE.TXT

LOAD, cargar en memoria

El comando no residente LOAD desarrolla la función opuesta a SAVE, ya que carga en la memoria central, siempre a partir de la posición 100H, el contenido del fichero especificado.

Es oportuno recordar que en el entorno CP/M los comandos no residentes están presentes sobre el disco como ficheros que tienen la extensión COM y que, cuando se invoca uno de estos comandos, el sistema carga el fichero requerido en la memoria central, partiendo de la posición 100H.

El comando LOAD tiene como efecto a veces el crear un nuevo comando no residente, ya que después de cargar el fichero especificado será suficiente pulsar el nombre del fichero, sin ninguna extensión, para que empiece a ser ejecutado: LOAD crea un fichero con extensión COM cuando el fichero cargado tiene una extensión HEX, es decir, cuando contiene las instrucciones de un programa en lenguaje máquina (Fig. 1).

Naturalmente, la orden dará buenos resultados sólo si el fichero requerido por LOAD contiene efectivamente un programa que se pueda ejecutar y que tenga alguna utilidad.

En resumen, la sentencia

A>LOAD NUEVOCOM.HEX

carga en la memoria central el conjunto de instrucciones en lenguaje máquina contenido en el fichero NUEVOCOM.HEX creando NUEVOCOM.COM.

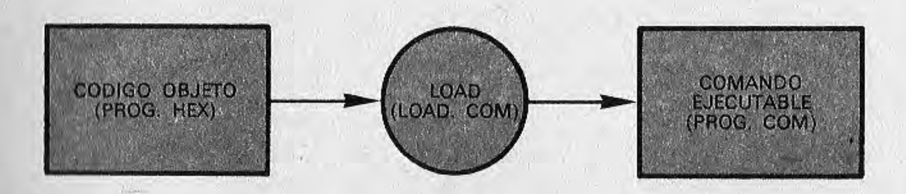


Figura 1.— Actuación del comando no residente LOAD.

Pulsando entonces

A>NUEVOCOM

se obtiene la ejecución efectiva de dichas instrucciones.

USER, divisiones en el disco

El CP/M ofrece la posibilidad, dentro de la gestión de un disco, de distribuir los ficheros en 16 áreas distintas, marcadas por los números 0 al 15, con el fin de permitir a los usuarios mantener separados ficheros de contenido o utilización diferente.

El comando USER es residente y, por lo tanto, utilizable siempre.

Si se desea una gran exactitud, el prompt del CP/M (nuestro típico A>) debería escribirse (o entenderse) como

ØA>

es decir, el área de usuario asociada es aquella marcada con el número Ø. Para pasar a un área diferente, por ejemplo 5, se debe pulsar:

A>USER 5

y la contestación será:

5A>

que ofrece al usuario la disponibilidad de los ficheros eventualmente presentes en el área 5. Si ahora realizamos un DIR nos mostrará solamente los ficheros presentes en este área.

El disco no está dividido en 16 partes iguales, para facilitar que el espacio disponible se pueda asignar a usuarios distintos, hasta un máximo de 16. La dimensión de cada área de usuario resulta dada dinámicamente por el volumen efectivo de los datos y por el número de áreas que coexisten hasta el agotamiento de todo el espacio.

Por lo tanto, es posible que sobre un disco existan, por ejemplo, solamente las áreas de usuario Ø,7 y 12, con una distribución de espacio totalmente desequilibrada.

Para tener una idea de cómo está dividido un disco en áreas de usuario se utiliza una función particular de STAT:

A>STAT USR:

cuya contestación especifica el área de usuario activa en ese momento y las áreas de usuario en las que existen ficheros, en la forma:

```
Active User: 0
Active Files: 0 7 12
A>
```

Como en el momento de la carga del S.O., el área enlazada es siempre la Ø, nace la necesidad del traslado de los ficheros a otras áreas. Este problema es resuelto por PIP con la opción Gn, donde "n" especifica el área de usuario del disco de la cual se debe obtener el fichero objeto de traslado; la copia se efectúa en el área de usuario activa en el disco de destino.

De todas maneras es necesario, para llevar a cabo la opción PIP, que el fichero PIP.COM esté presente en el área de usuario de destino. Si no es así, lo primero será trasladarla.

Examinemos la secuencia necesaria para trasladar PIP.COM al área de usuario 4:

A>USER 0	se parte del área Ø.
A>DDT PIP.COM	se carga en memoria PIP (ver la función de DDT en el cap. 7).
DDT vers. xxx.xx	
NEXT PC	
1C80 xxxx	
-GO	se vuelve al CP/M.
A>USER 4	se vuelve activa el área 4
4A>SAVE 28 PIP.COM	se traslada PIP.COM desde la memoria al área 4
4A>	

y la secuencia para copiar el fichero MUESTRA del área 2 del disco B al área 4 del disco A es:

```
4A>PIP A:=B:MUESTRA (G2)
```

Es posible copiar solamente en el área de usuario activa, pero no hace falta utilizar el parámetro G cuando el área de usuario

activa tiene el mismo número que aquella de la que se debe obtener el fichero.

La utilización de USER permite mantener efectivamente separados sobre disco datos de distinta naturaleza. Por experiencia, les aconsejamos no abusar de esta función.

ASM, un lenguaje a adoptar

Una manera de controlar las posibilidades de una computadora es crear secuencias de órdenes en lenguaje máquina que manejen el microprocesador de la manera más rápida posible.

Existen funciones que el microprocesador desarrolla rápidamente, y que deben respetar unas convenciones. El conjunto de estas reglas sintácticas permite la redacción de programas en lo que se llama lenguaje Ensamblador (Assembler).

Para transformar estos programas en programas objeto, ya en lenguaje máquina, se utiliza el comando no residente ASM.

Este comando lleva a cabo la función de traducir, según el contenido de una tabla interior, todas las instrucciones contenidas en un fichero con extensión ASM a las correspondientes expresiones, compatibles con la sintaxis del microprocesador 8080; después las guarda en un fichero del mismo nombre, pero con extensión HEX, y crea un fichero con extensión PRN que contiene información auxiliar de la traducción efectuada (Fig. 2).

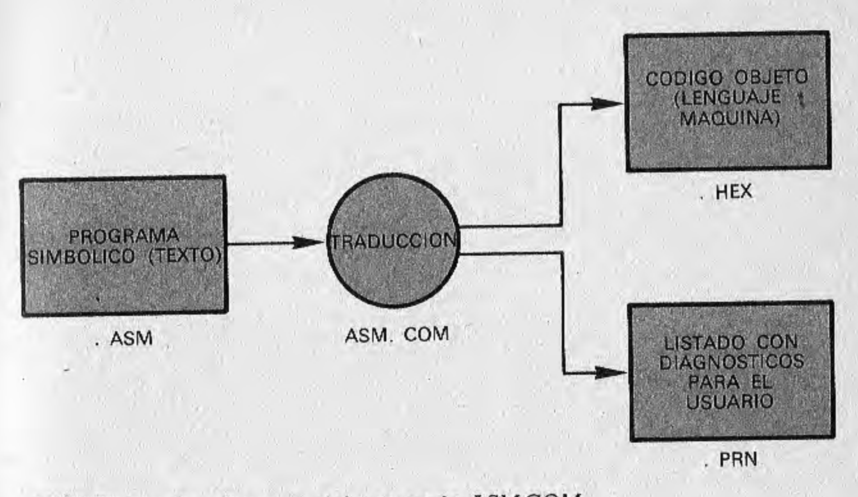


Figura 2.— Proceso del comando ASM.COM.

La sintaxis del ASM permite especificar un grupo de tres parámetros, que indican, respectivamente: la unidad sobre la cual operar para el control del fichero de origen, del que contiene la traducción en código hexadecimal y del que contiene la información auxiliar.

La sintaxis es:

ASM nombre-de-fichero.shp

shp son los tres parámetros que mencionamos anteriormente.

- S letra identificadora de la unidad que contiene el fichero .ASM origen (A, B, ... P).
- h letra de la unidad que recibirá al fichero .HEX, creado por el ASM (A, B, ... P)
- p Puede ser:
  - — la letra de la unidad (A, B, ... P) que recibirá el fichero .PRN creado;
  - — "X", si el fichero se manda al terminal;
  - — "Z", si no se crea el fichero.

La utilidad de estos parámetros se encuentra de nuevo en las fases preliminares de la preparación del programa: muy a menudo es útil llevar a cabo una rápida "vuelta" de compilación para averiguar si la sintaxis del programa es correcta y evitar los posibles mensajes de error que el traductor envía. Sólo cuando el programa está casi perfectamente a punto será oportuno llevar a cabo el procedimiento entero que, por el hecho de tener más fases y funciones, es necesariamente más lento.

En caso de que se quiera transformar el PROGRAM.ASM se deberá pulsar:

A>ASM PROGRAM.BAX

de donde el S.O. deducirá que:

- el fichero PROGRAM.ASM (que contiene las instrucciones a traducir) está en el drive B (del primer parámetro);
- el fichero PROGRAM.HEX (que contiene los códigos determinados por la traducción) se situará en el drive A (del segundo parámetro);
- el fichero PROGRAM.PRN (subproducto de la traducción) se visualizará, pero no se guardará sobre disco (del tercer parámetro).

En el caso en que la unidad A sea activa, resultarán equivalentes las expresiones:

A>ASM PROGRAM.AAA
A>ASM PROGRAM

La lista de las normas que determinan la sintaxis de las instrucciones admitidas se aleja del tema, pero se puede encontrar fácilmente en cualquier documentación gracias a la enorme difusión de los sistemas basados sobre micros de la familia del 8080.

XSUB, una reclamación necesaria

Una función útil que cumple el comando SUBMIT es la de actuar de manera generalizada, permitiendo que la especificación de los parámetros se efectúe en el momento de la ejecución. Esto permite crear unas rutinas muy flexibles.

Vamos a realizar una que ensamble un fichero, imprima el fichero relativo (PRN) y lo destruya al final de la sesión.

La ventaja que deseamos obtener es no tener que especificar nada más que el nombre del fichero en el momento del lanzamiento de la macro y poder elegir el imprimirle o leerle en la pantalla.

Prácticamente se crea un fichero SUB en el que se indica de manera genérica cada dato a especificar sucesivamente. Denominaremos ENSAMBLA.SUB al fichero:

```
ASM $1
PIP $2:=$1.PRN
ERA $1.PRN
```

$1 y $2 que, como vemos, pueden aparecer como $1 más veces, son en la sintaxis de la orden el primero y el segundo parámetro.

En el momento de la ejecución solamente se debe especificar el nombre del fichero a ensamblar y el dispositivo por el que sacar el PRN.

La orden, tomando MUESTRA.ASM y PRN es:

A>SUBMIT ENSAMBLA MUESTRA PRN

La secuencia efectiva de las órdenes será:

```
ASM MUESTRA
PIP PRN:=MUESTRA.PRN
ERA MUESTRA.PRN
```

El efecto se cumple al sustituir los nombres colocados después del SUBMIT ENSAMBLA a los correspondientes símbolos $n ordenadamente. En nuestro ejemplo MUESTRA se ha trasladado al sitio de $1 y PRN al de $2.

El comando SUBMIT permite tomar órdenes de un fichero y no de la consola, pero esta función está limitada a los que no requieren, por su naturaleza, la intervención de procedimientos particulares del S.O.

En el ambiente CP/M existen comandos que redactan líneas de texto o ficheros enteros (ED y DDT) que, por su funcionamiento, modifican la estructura normal del S.O. para permitir la utilización de órdenes particulares.

En un fichero SUB una orden como ésta (por ejemplo L, A, R, etc.) no podrá ser aceptada, ya que el sistema considera cada línea compuesta por una orden independiente y de sintaxis CP/M tradicional.

La única manera de resolver este problema es introducir, como primera línea del fichero SUB, el comando no residente XSUB (XSUB.COM deberá estar en el disco), que comunica al CP/M que en las líneas siguientes están contenidas secuencias de caracteres que normalmente sólo se podrían enviar en presencia de funciones particulares.

Utilicemos esta función para crear un pequeño programa que organice un fichero de texto en páginas y permita su visualización a partir de una línea dada.

La rutina llamada EJEMPLO.SUB explota unas opciones del comando ED que permiten dividir un fichero en páginas de 23 líneas y visualizarlas a partir de la línea elegida.

Su contenido será:

```
XSUB
ED $1
#A
$2OP
```

y los parámetros a sustituir son sencillamente: el nombre del fichero a leer ($1) y el número de línea desde la cual deberá partir la visualización ($2). Como ejemplo leeremos el fichero MUESTRA.TXT a partir de la trigésimoquinta línea:

```
A>SUBMIT EJEMPLO MUESTRA.TXT 35
```

Cuando acabe la primera página aparecerá el número de la primera línea visualizada y un asterisco. Si pulsamos P (y Return) se visualizará la página siguiente, y si hacemos lo propio con -P será la página anterior la mostrada. Tal y como especifica la sintaxis del comando ED (que veremos) el pulsar la E será entendido como el deseo de volver al control del S.O.

Con un poco de práctica e imaginación podrá sacar rendimiento a las macros que cree con SUBMIT y XSUB.

SYSGEN, cargar y copiar el sistema

Un disco que contenga los ficheros del sistema se podrá utilizar para cargar el CP/M en la memoria de la computadora, mientras un disco que no los contenga solamente podrá ser utilizado después de esta operación.

Como estos ficheros no residen en partes accesibles del sistema, no están sujetos a las órdenes normales tipo PIP o ERA. La única manera en que pueden ser trasladados a un disco y, por lo tanto, crear un disco con el sistema es utilizando una función específica muy importante llamada SYSGEN, que significa "Sistem Generate".

SYSGEN traslada los ficheros del sistema al área de memoria que parte de la posición 100H, permitiendo, por lo tanto, el traslado sucesivo al disco deseado.

Cuando se recibe el disco del sistema, al comprar el ordenador es conveniente hacer una copia del mismo y guardar el original en sitio seguro; de este modo, si por cualquier motivo se estropeara la copia, siempre podríamos recurrir al original.

Examinemos la sucesión de los mensajes que aparecen en la pantalla al pulsar

```
A>SYSGEN
SOURCE DRIVE NAME (OR RETURN TO SKIP)?
```

en este momento el usuario debe comunicar al sistema el drive (la unidad) que contiene el disco desde el que se puede cargar el S.O. En el caso de que el sistema haya sido cargado ya en memoria, será suficiente pulsar RETURN para saltar las fases referidas a la carga.

Por ejemplo, pulsando A, la contestación es:

```
SOURCE ON A THEN TYPE RETURN
```

Estas palabras invitan al usuario a introducir en el drive A el disco que contiene el S.O. antes de volver a pulsar RETURN.

El mensaje siguiente será:

```
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT)?
```

que pregunta al usuario el drive en que está contenido el disco en el cual debe copiar el S.O.

Siguiendo con el ejemplo se pulsa B, obteniendo:

DESTINATION ON B THEN TYPE RETURN

mensaje de espera para permitir la colocación del disco en el drive B. La pulsación de RETURN producirá:

FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT)?

que permite una nueva copia y otras sucesivas hasta que el usuario transmita su intención de terminar pulsando sólo RETURN.

La copia del S.O. no daña la integridad de los ficheros que ya estén en el disco.

MOVCPM, el milagro de la autorresolución

Ya hemos hablado de la flexibilidad del CP/M en cuanto a aceptar y controlar modificaciones sobre sí mismo.

La posibilidad de volver a definir una versión del CP/M para que se adapte a diferentes capacidades de memoria se puede explotar por medio del comando no residente MOVCPM. Éste hace todas las transformaciones necesarias para crear una nueva versión del CP/M que tenga en cuenta la estructura y capacidades del ordenador en que va a operar. Para cargar el CP/M en la memoria y hacerle obedecer a la orden MOVCPM es necesario activar la orden SYSGEN y utilizar de ella sólo la primera parte, pulsando RETURN a la pregunta "DESTINATION DRIVE NAME".

El comando MOVCPM admite dos parámetros cuyo significado es establecer las dimensiones a que debe adaptarse el nuevo sistema y abandonar o no la parte de memoria ocupada por la conversión, al final del proceso de transformación y ejecución.

Al primer parámetro se le puede atribuir un valor que va desde 20 hasta 64 kilobytes; si se especifica "*", se entiende la máxima dimensión permitida por el sistema que lo recibe.

El segundo parámetro puede ser un asterisco o bien no estar presente. En el primer caso la nueva versión del CP/M quedará en memoria en espera de ser trasladada a un disco, mientras en el segundo hará lo mismo, pero dejará libres las posiciones ocupadas al terminar la ejecución.

La imagen del S.O. se localizará en la zona de memoria incluida entre las posiciones 900H y 227FH. Por lo tanto:

A>MOVCPM 48*

realizará una versión del CP/M apta para una capacidad de 48K, la guardará en memoria y visualizará el mensaje

READY FOR 'SYSGEN' OR 'SAVE 34 CPM48.COM'

que nos indica los dos posibles pasos siguientes:

a) Nueva llamada al comando SYSGEN, del cual se saltará la primera parte (pulsando RETURN al requerimiento de especificar el "SOURCE DRIVE NAME", lo cual permitirá trasladar a un disco la nueva versión realizada, o también

b) la utilización de la función

SAVE 34 CPM48.COM

que salvará en el disco las primeras 34 páginas de memoria con el nombre CPMxx.COM (en este caso xx equivale a 48) para permitir posteriores modificaciones del sistema.

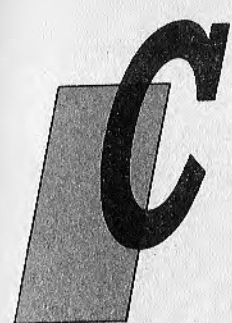
La orden MOVCPM crea una versión del S.O. válida para una capacidad de memoria distinta.

No se debe olvidar que la modificación del número o de las características de los periféricos hace necesaria una intervención directa en la sección "Hardware-dependiente" del CP/M, el BIOS.

CAPITULO VI

EDITOR DE LÍNEA: EL COMANDO ED

Principios básicos de funcionamiento



on mucha frecuencia, el coloquio que mantiene el operador con el sistema requiere la escritura, composición y control de los textos.

Los programas para la redacción de textos toman el nombre, como se sabe, de WORD PROCESSOR (o TEXT EDITOR) y presentan una amplia gama de modelos: desde los más sofisticados (utilizados en las oficinas) a los más sencillos, de uso personal.

En el caso de que la utilización se limite a la composición de ficheros SUB o de cortos programas urgentes en Ensamblador no se justifica la adquisición de un evolucionado instrumento de WORD PROCESSING, a veces muy caro.

Este es el motivo de la utilización de un editor como la función ED que, aunque un poco complicada, resuelve los problemas más sencillos.

El objeto del comando ED está constituido por un solo fichero. Por lo tanto,

A>ED

no tiene ningún significado.

Su sintaxis correcta es:

ED nombre-de-fichero.ext

Si ED no encuentra el fichero especificado, asume que no se trata de la modificación de un fichero, sino de la creación de uno nuevo.

ED, de todas formas, no actúa directamente sobre los caracteres que constituyen el fichero original, ya que crea un duplicado temporal de éste.

La versión original del fichero no se pierde, pues se le impone una extensión especial, la extensión BAK, con el fin de conservarlo como copia de seguridad (backup) para el caso en que sea imposible terminar la sesión de edición del nuevo fichero de manera correcta.

Las modificaciones se realizan en la memoria temporal (de tránsito o buffer).

Los caracteres correctos se transfieren seguidamente a un fichero de servicio con la extensión $$$. Solamente al final de la sesión de edición se da al fichero $$$ el nombre definitivo especificado y la memoria de tránsito queda libre (Fig. 1).

La estructura general del comando ED, en términos de espacio de memoria empleado, resulta esquematizada en la figura 2, a la cual el lector deberá hacer referencia en el curso de la exposición que sigue.

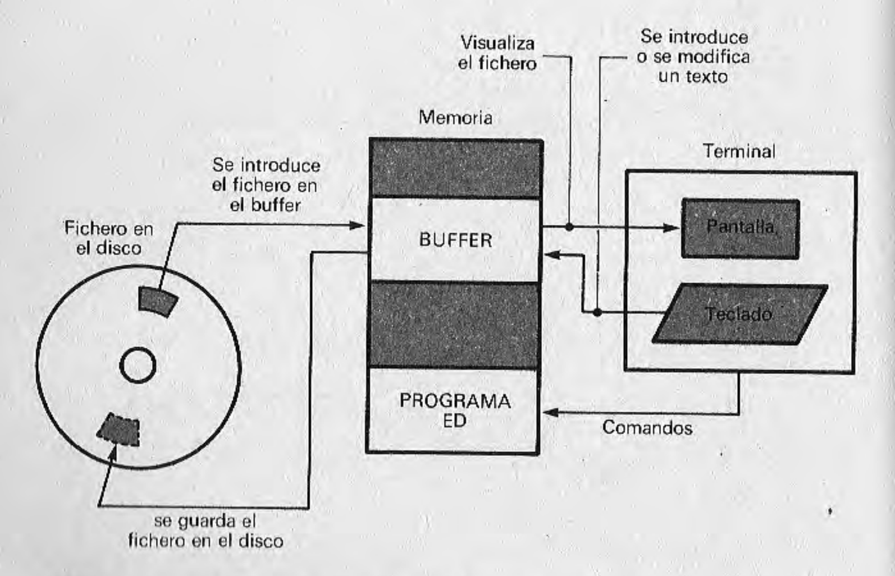


Figura 1.— Movimientos de memoria con el comando ED.

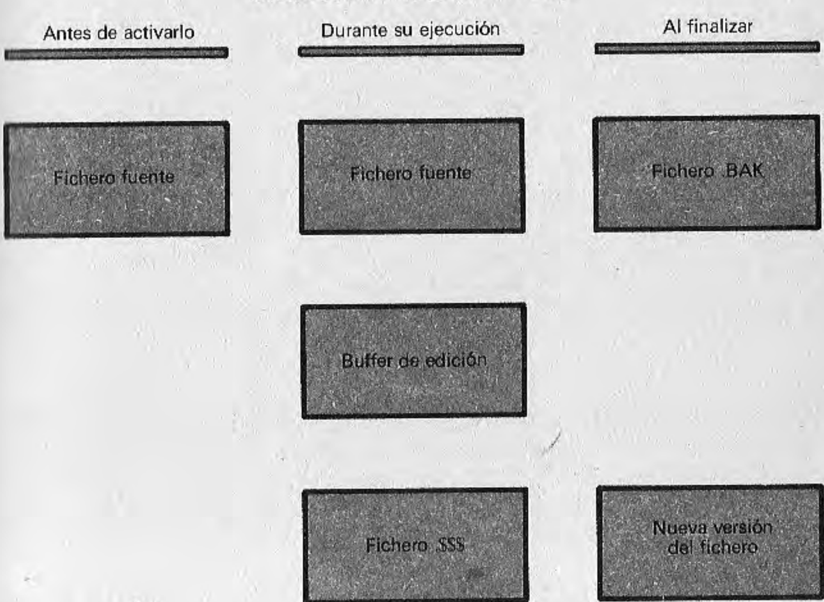


Figura 2.— Estructura general del comando ED expresada en términos de la situación de la memoria antes, durante y después de la actuación del comando.

El envío de la orden

A>ED EDICION.TXT

(nombre que se utilizará en todos los ejemplos) produce la aparición del mensaje

NEW FILE
:*-

NEW FILE comunica que el sistema ha examinado todo el directorio del disco en el drive A sin encontrar un fichero con ese nombre, por lo que interpreta que es nuevo y procede a crearlo; seguidamente se prepara para guardar en el fichero de nombre EDICION.TXT recién creado todo lo que el usuario le indique. :*- es, en cambio, el prompt de la función ED, la señal de espera de órdenes.

Como estamos trabajando en un fichero nuevo, no tiene sentido enviar órdenes de lectura, modificación o cancelación de caracteres. La única alternativa válida es, por tanto, introducir texto. Pulsemos entonces

:*i

(i de inserción) seguido de RETURN. La contestación es

1:-

con lo que ED nos indica que considerará todos los caracteres pulsados como parte de la primera fila del fichero.

Las órdenes del comando ED se escriben en letras mayúsculas cuando se quieren transformar en mayúsculas todos los caracteres pulsados y en minúsculas cuando los caracteres se deben interpretar tal y como están escritos.

La escritura de una fila mantiene las mismas características que se pueden desarrollar con una simple máquina de escribir, aunque con estas limitaciones:

- el número máximo de caracteres que se pueden introducir en una fila es 48;
- al final de cada fila se debe pulsar RETURN;
- el número máximo de filas que se pueden introducir es 65.536.

Para terminar la inserción bastará pulsar CTRL+Z (^Z). El efecto consistirá en presentar de nuevo el prompt:

:*-

es decir, poner de nuevo al ordenador en espera de órdenes. Conviene crear un fichero de prueba en el que podemos introducir:

1: Nos preparamos a redactar un fichero
2: que nos permitirá comprobar ↵
3: todos los instrumentos ↵
4: proporcionados por ED. ↵
5: ^Z
:*-

Si pulsamos la letra "q" (de "quite", abandono) y RETURN, el programa responderá

Q-(Y/N)?

es decir, la confirmación de la petición de abandono. Pulsando Y el usuario afirma su intención de abandonar lo que ha hecho, renunciando en nuestro caso a la creación del fichero EDICION.TXT. Pulsando N visualizaremos nuevamente el prompt de espera de órdenes.

Si escribimos

:*e

(de "exit", salida) tiene el efecto de terminar la edición, grabar en disco los caracteres introducidos (con el nombre EDICION.TXT) y visualizar el prompt del sistema operativo de nuevo:

A>

es decir, sale del programa ED para volver al CP/M. En el caso que sea necesario modificar el contenido del fichero será posible volverlo a llamar de la misma manera.

Vamos a ver cómo se pueden llevar a cabo modificaciones en un fichero ya existente. La orden necesaria es:

A>ED EDICION.TXT

pero también se admite la posibilidad de cambiar el nombre al fichero. Se actuaría en ese caso así:

A>ED EDICION.TXT B: NUEVO.DOC.

La orden crea en el disco B un nuevo fichero llamado NUEVO.DOC que contiene el resultado de las correcciones en EDICION.TXT. De esta manera ED lleva a cabo solamente tareas de preparación, ya que dispone un área de la memoria que recibirá las líneas a modificar y crea un fichero temporal de salida con la extensión $$$, reservado a las líneas ya corregidas. En este momento en el disco existen dos ficheros:

— EDICION.BAK, el fichero original, y
— EDICION.$$$, el fichero virgen que representará, al final de las correcciones, la nueva versión de EDICION.TXT.

Es obligatorio ahora comunicar al ED cuántas son las líneas del fichero sobre el cual se deberá operar.

Hay que subrayar que el número que identifica las líneas no es parte integrante de las mismas, sino sólo un artificio para la grabación ordenada de los datos. En el caso de que se desee trabajar en tres líneas, deberemos pulsar:

:*3a

y en seguida las primeras tres líneas del fichero serán trasladadas a un área particular de la memoria, llamada buffer de edición (Fig. 1). Solamente el contenido de este buffer se puede corregir. De hecho, es más corriente que el usuario desee operar sobre todo el fichero y no tenga ningún interés en limitar su dimensión efectiva.

Lo mejor, por tanto, es reservar todo el buffer disponible, especificando la máxima dimensión que un fichero puede alcanzar, es decir 65.536 líneas.

Como muy frecuentemente se prefiere la dimensión máxima, ED utiliza el símbolo "#" para ese fin.

:*#a

indica, por tanto, que deseamos usar todo el buffer disponible y nos permite actuar sobre cada uno de los caracteres del fichero.

El buffer está organizado con una lógica basada en el control de cada carácter; por lo tanto, el usuario, para introducir nuevas frases, deberá especificar dentro de qué línea y antes de qué carácter debe ser realizada la inserción.

En cambio, para llevar a cabo una sustitución se debe indicar en qué línea y a partir de qué carácter tendrá que efectuarse la modificación.

Para alcanzar este resultado de manera eficiente ED controla el movimiento de un puntero de los caracteres que componen las líneas.

En el momento de volver a nombrar un fichero, el puntero (o pointer, término que deriva más de la balística que de la raza canina) se situará antes del primer carácter de la primera línea. Un rápido desplazamiento del puntero se consigue con las órdenes B (que lo lleva al principio del fichero) y -B (que lo lleva al final del mismo).

Normalmente se necesita leer el contenido del fichero antes de corregirlo. El comando nT permite visualizar el contenido de las "n" líneas que siguen al puntero de caracteres.

Un método a seguir cuando se empieza a corregir el fichero es pulsar la secuencia

```
A>ED nombre-del-fichero.extensión
:*#a
1:*b#t
```

La última orden provoca la visualización continua del contenido del fichero, que se puede interrumpir pulsando una tecla cualquiera.

Movimiento del texto

Hemos visto el mecanismo por el que se rige el funcionamiento de ED. Examinamos ahora detalladamente el efecto de los distintos comandos de movimiento del texto hacia y desde el buffer.

nA transfiere "n" líneas desde el fichero de origen (o fuente) al buffer de edición.

Para permitir la corrección de ficheros de grandes dimensiones, que podrían no caber en el buffer puesto a nuestra disposición en la memoria, se puede utilizar esta función en varios momentos. ED controlará, en el fichero origen, un puntero de línea que lleve la cuenta de las líneas ya trasladadas y otro puntero del espacio ocupado en el buffer. El valor máximo que puede tomar "n" es 65.535, mientras A sólo especifica una línea y #A el fichero entero.

Una utilización especial está reservada a 0A, que produce el efecto de cargar en el buffer líneas en número no superior a la mitad del mismo buffer. Este mecanismo previene el desbordamiento del buffer en el caso de ficheros grandes. Siguiendo con el ejemplo:

```
:*#a
:*_
```

trasladará todo el fichero EDICION.TXT al buffer de edición.

nW es la orden opuesta a A, en cuanto que traslada "n" líneas desde el buffer al fichero temporal, actualiza el puntero de las líneas del buffer, trasladándolo hacia atrás "n" líneas, y mueve "n" líneas hacia adelante el puntero de líneas del fichero temporal.

El valor máximo alcanzable por "n" es 65.535. W, por sí solo, traslada una única línea, mientras #W pasa el buffer entero al fichero temporal.

La orden 0W traslada al fichero temporal todas las líneas necesarias para que medio buffer quede vacío.

El uso conjunto y repetido de 0A y 0W permite la edición de ficheros de grandes dimensiones.

Ejemplo:

```
:*#W
:*_
```

su efecto será vaciar completamente el buffer a beneficio de EDICION $$$.

E termina la sesión de corrección y copia todo el buffer en el fichero temporal $$$.

Si en el fichero origen existen otras líneas que no han sido trasladadas al buffer, E las lleva al fichero temporal sin ninguna modificación.

El fichero $$$ tiene el nombre elegido en el momento de activar ED y el fichero origen toma la extensión BAK.

El buffer se borra totalmente y volvemos al S.O. con la visualización del prompt A>:

```
:*e
A>
```

El efecto es la cancelación del buffer, cambiando el nombre del fichero EDICION.TXT por EDICION.BAK y el de EDICION.$$$ por EDICION.TXT.

H lleva a cabo las funciones de E, es decir de final de la sesión, pero no vuelve al CP/M, sino que activa de nuevo el ED sobre el mismo fichero.

Esta función salva todas las modificaciones efectuadas y permite empezar de nuevo la edición sobre un fichero origen más actualizado.

Naturalmente, el buffer y el fichero temporal comienzan completamente vacíos y los punteros de líneas de los ficheros origen y temporal se colocan al inicio de los respectivos ficheros.

Ejemplo:

```
:*h
:*-
```

El efecto es borrar el buffer, cambiar el nombre del fichero EDICION.TXT por EDICION.BAK y EDICION.$$$ por EDICION.TXT, destruir EDICION.BAK, cambiar el nombre del nuevo EDICION.TXT por EDICION.BAK y abrir un nuevo fichero EDICION.$$$.

Q anula todas las correcciones efectuadas y devuelve al usuario al CP/M. Esta orden no modifica absolutamente en nada al fichero origen y destruye el fichero temporal y el contenido del buffer.

Ejemplo:

```
:*q
Q-(Y/N)? Y
A>
```

El efecto consistirá en volver a llevar el nombre de EDICION.BAK a EDICION.TXT y destruir EDICION.$$$.

O produce los mismos efectos que Q, es decir, anula toda modificación efectuada, pero activa de nuevo ED con el mismo fichero.

En la práctica pone a cero el buffer y el fichero temporal, llevando los punteros a las posiciones iniciales. O, por lo tanto, actúa como H, pero sin llevar a cabo las modificaciones.

Control del buffer

El buffer se controla actuando sobre cada carácter, lo que implica que todas las modificaciones deben de especificar el carácter antes o después del cual se debe efectuar la acción deseada.

Es necesario, por lo tanto, conocer perfectamente cómo el puntero de caracteres se mueve en el buffer. Hay que tener en cuenta que el movimiento del puntero no produce la visualización del carácter sobre el que se coloca. Si añadimos que en el interior del buffer el paso al principio de la línea siguiente se traduce en una secuencia de dos órdenes (retorno de carro y nueva línea) vemos que las dificultades aumentan.

Todo esto obliga al usuario a abusar del comando T de visualización y a realizar complejas secuencias de órdenes con el fin de obtener la cuenta de los caracteres y líneas por los que se ha movido el puntero.

De hecho el puntero no se sitúa sobre el carácter, sino antes de éste. Es decir, separa netamente el conjunto de los caracteres que le preceden y aquél de los caracteres que le siguen.

Cuando el puntero se sitúa al inicio de una línea está colocado delante del primer carácter de ésta.

Veamos detalladamente las órdenes de control del puntero:

n: es la orden que permite mover el puntero al inicio de la línea cuyo número es "n". Este número es controlado por el mismo buffer y es utilizado sólo en la sesión de edición.

En caso de que el usuario tenga la necesidad de con-

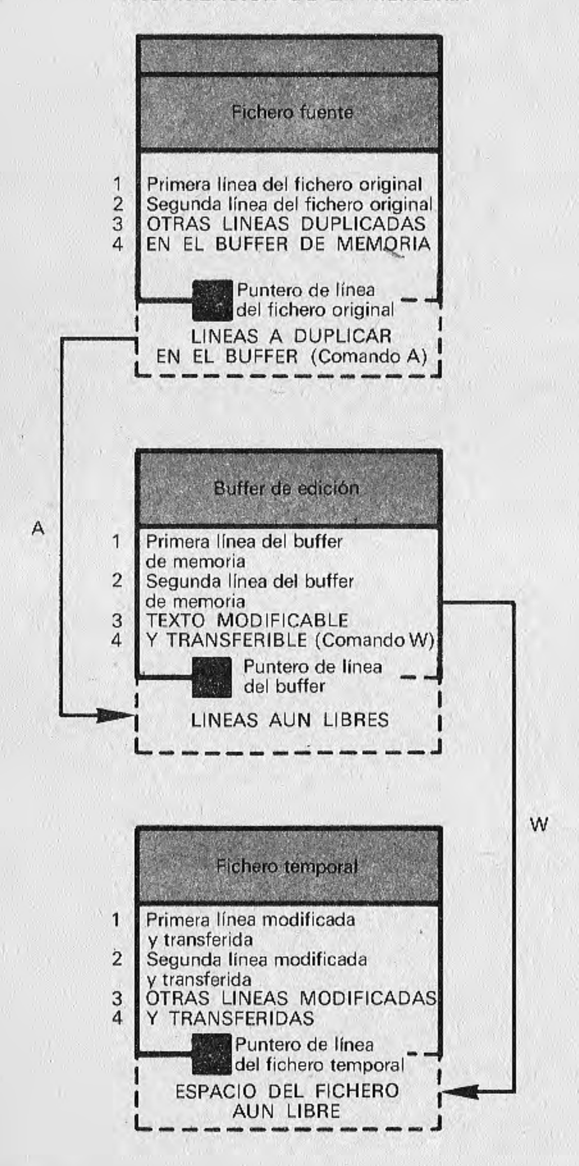


Figura 3.— Organización de la memoria tal y como es controlada por el comando ED, incluyendo el juego de los punteros.

trolar los números de línea puede usar el comando V (y anularlo con -V).

El valor máximo de "n" es 65.535.

Ejemplo:

```
:*3:
3:*-
```

El efecto consiste en mover el puntero sin ninguna visualización.

B — mueve el puntero al inicio del buffer (antes del primer carácter).

-B — mueve el puntero al final del buffer (después del último carácter contenido en el buffer)

```
3:*b
1:*-
```

coloca el puntero al inicio del fichero, es decir, antes del primer carácter de la línea 1.

nC — mueve el puntero hacia adelante "n" caracteres (cuidado al hecho de que cada vuelta al principio de la línea siguiente está representada por dos caracteres).

-nC — lleva atrás "n" caracteres el puntero.

Ejemplo:

```
1:*b
1:*18c
1:*t
mostrar un fichero
1:*-
```

Al requerimiento de visualización de la línea (T) ED ha contestado enseñando desde el decimoctavo carácter hasta el final de la línea, ya que 18C ha movido el puntero más allá de "Nos preparamos a ".

nD — borra los "n" caracteres que siguen al puntero.

-nD — borra los "n" caracteres que preceden al puntero.

Ejemplo:

```
1:*b25c3dbt
1: Nos preparamos a mostrar fichero
1:*-
```

con la primera secuencia de órdenes hemos movido el puntero al inicio del buffer (B), luego lo hemos situado inmediatamente antes de " un" (25 C), borrado estos tres caracteres (3 D), vuelto al inicio del fichero (B) y, finalmente, hemos visualizado la línea que contiene el puntero (T). Como se puede comprobar han desaparecido los tres caracteres no deseados ("u", "n" y " ").

nK borra las "n" líneas que siguen al puntero. En caso de que el puntero no estuviera al inicio de una línea, los caracteres que le preceden no se destruyen.

-nK borra las "n" líneas que siguen al puntero. En caso de que el puntero no esté al inicio de una línea, los caracteres que le siguen no se destruyen.

Ejemplo:

```
1:*2:2Kb#t
1: Nos preparamos a mostrar fichero
2: Creado por ED.
1:*-
```

La secuencia ha determinado el desplazamiento del puntero al inicio de la segunda línea (2:), la cancelación de las dos líneas sucesivas (2K borra la segunda línea, donde se encuentra el puntero, y la tercera), el desplazamiento del puntero al inicio del buffer (B) y la visualización de todo el buffer (#T).

nL lleva el puntero al inicio de la línea donde se encuentra y lo mueve "n" líneas hacia adelante. ØL efectúa solamente el desplazamiento del puntero al inicio de la línea corriente. L equivale a 1L.

-nL lleva el puntero al inicio de la línea donde se encuentra y lo desplaza hacia atrás "n" líneas.

Ejemplo:

```
:*blt
2: Creado por ED.
2:*-
```

La secuencia tiene el efecto de desplazar el puntero al inicio del buffer (B) y luego llevarlo al inicio de la línea siguiente (L) y visualizarlo (T).

nP desplaza el puntero hacia delante 23 líneas y visualiza las 23 líneas siguientes; repite esto "n" veces. ØP muestra solamente las 23 líneas que siguen al puntero, pero no lo desplaza.

-nP lleva hacia atrás el puntero 23 líneas y visualiza las 23 líneas siguientes; lo repite "n" veces.

nT visualiza "n" líneas a partir del primer carácter que sigue al puntero. ØT visualiza todos los caracteres desde el puntero hasta el final de la línea corriente.

-nT visualiza las "n" líneas que preceden al puntero.

n equivale a la secuencia nLT que mueve el puntero "n" líneas hacia adelante y enseña la línea sobre la cual se ha desplazado.

-n equivale a la secuencia -nLT, que lleva el puntero hacia atrás "n" líneas y visualiza las líneas sobre las cuales se ha desplazado.

n:T lleva el puntero sobre la línea "n" y la visualiza.

:nT visualiza todas las líneas a partir del puntero hasta aquella con número "n".

r1::r2T mueve el puntero al inicio de la línea con número r1 y visualiza todas las líneas desde r1 a r2.

Ejemplo:

```
:*1: :2t
1: Nos preparamos a mostrar fichero
2:Creado por ED
1:*-
```

U transforma todas las letras minúsculas en las correspondientes mayúsculas.

-U anula la transformación de minúsculas en mayúsculas.

V verifica la numeración de las líneas.

-V anula la numeración de las líneas.

ØV visualiza la relación entre el número de bytes libres del buffer y el máximo número de bytes que puede admitir con el formato:

bytes-libres/dimensión-buffer

nZ interrumpe la ejecución del programa unos "n" segundos.

Órdenes inmediatas con CTRL

En el curso de las modificaciones llevadas a cabo sobre los caracteres del fichero origen es posible aprovechar un conjunto de órdenes especiales que no necesitan de la pulsación de la tecla RETURN para ser cumplimentadas.

Estas órdenes, peculiares del CP/M, están todas compuestas por la famosa tecla CONTROL (CTRL) y una letra, y producen inmediatamente el efecto deseado.

Veamos cuáles son:

^C interrumpe la ejecución de cualquier orden y vuelve al CP/M.

^E reproduce el carácter de final de línea. Esta acción permite continuar con la introducción de caracteres en la línea siguiente hasta que se pulse la tecla RETURN.

^H borra el último carácter pulsado.

^J reproduce el carácter de final de línea e interrumpe la introducción de caracteres.

^L representa el final de la línea. Está formado por dos caracteres, retorno de carro y línea nueva.

^M reproduce el carácter de retorno de carro e interrumpe la introducción.

^R visualiza la línea actual.

^U borra la línea pulsada.

^X borra la línea pulsada.

^Z establece el final de la introducción de caracteres.

Órdenes avanzadas: facilitándonos el trabajo

Por la complejidad del mecanismo con el que el puntero individualiza la posición sobre la cual efectúa las funciones deseadas, ha sido necesario dotar a ED con órdenes más evolucionadas que ahorrasen al usuario las tareas más duras.

La lógica que anima estas órdenes es la individualización de una cadena particular, especificada en la línea de comando, a partir de la cual se llevan a cabo las modificaciones oportunas.

La búsqueda de la posición de una cadena (F=Find, encontrar) se lleva a cabo con:

nfcadena^Z donde, como siempre, ^Z indica el final de la cadena y...

-n especifica cuántas veces se debe llevar a cabo la búsqueda; en cuanto al puntero, se colocará detrás de la enésima ocurrencia de la cadena buscada.

Si la cadena no es hallada, el puntero no se desplaza.

Sigamos utilizando el ejemplo de los párrafos anteriores:

```
1:*bffichero^Z-16ct.
mostrar fichero
1:*-
```

La secuencia ejecutada por esta línea será:

B desplaza el puntero al principio del buffer,

fichero^Z localiza la cadena "fichero" y coloca el puntero tras ella,

-16C lleva el puntero 16 caracteres hacia atrás, o sea antes de la palabra "mostrar".

T visualiza todos los caracteres comprendidos entre el puntero y el final de línea.

La inserción de una cadena se efectuará con:

icadena^Z recuerde que es necesario utilizar las iniciales minúsculas de los comandos si no se desea la transformación automática de las letras minúsculas en mayúsculas.

1:*bfichero^Z-7Citodos los ZØlt
1: Nos preparamos a mostrar el primer fichero
1:*-

La composición de la orden es:

ffichero^Z	localiza la cadena "fichero" y coloca el puntero tras ella,
-7C	desplaza el puntero 7 caracteres a la izquierda, o sea antes de la palabra "fichero",
i el primer^Z	introduce inmediatamente después del puntero la cadena "el primer",
ØL	desplaza el puntero al principio de la línea,
T	visualiza el contenido de la línea.

Los dos comandos anteriores se pueden fundir entre sí, o bien ser sustituidos por el comando (S=Substitute):

nscadena1^Zcadena2^Z

tiene el efecto de localizar la cadena 1, desplazarse hasta su principio, borrarla e introducir la cadena 2; repite esta secuencia "n" veces.

1:*sfichero^Zel primer fichero^ZØlt
1: Nos preparamos a mostrar el primer fichero
1:*-

secuencia que produce el mismo efecto que la anterior.
Estas órdenes tienen efecto en el ámbito del buffer, es decir, ejecutan la búsqueda sólo en el interior de las líneas memorizadas.
Si se quiere extender la búsqueda a todo el fichero origen se utilizará la orden N y se deberá pulsar

n1 ncadena^Z

que produce los mismos efectos que n1F si la búsqueda se delimitara al ámbito del buffer. Si, en cambio, la búsqueda debe realizarse en el fichero origen, todo el contenido del buffer es trasladado al fichero temporal para hacer sitio a las nuevas líneas a examinar hasta que la mitad del buffer esté lleno.

La búsqueda proseguirá indefinidamente hasta la completa satisfacción de la orden o hasta el agotamiento del fichero.
Otra orden es la que lleva a cabo la yuxtaposición. Su sintaxis es:

njcadena1^Zcadena2^Zcadena3^Z

el sistema busca "n" veces la cadena1, coloca el puntero inmediatamente a su final, introduce la cadena2, desplaza el puntero inmediatamente después de ella y borra todos los caracteres que siguen al puntero y preceden a la cadena3. El puntero queda siempre tras la cadena2 y ninguna cancelación es ejecutada si la cadena3 no es localizable.

1:*bjmostrar ^Z algunos de los ficheros ^Z ^L ^ZØlt
1: Nos preparamos a mostrar algunos de los ficheros
1:*-

La secuencia se interpretará como

B	desplaza el puntero al principio del buffer;
jmostrar ^Z	desplaza el puntero detrás de la cadena "mostrar";
algunos de los ficheros	introduce la cadena "algunos de los ficheros" y coloca el puntero tras ella,
^L ^Z	destruye todos los caracteres comprendidos entre el puntero y la cadena ^L, o sea el final de la línea;
ØL	desplaza el puntero al principio de la línea;
T	visualiza el contenido de la línea.

Para todas las órdenes de este párrafo valen las siguientes normas:

— No se admiten más de 100 caracteres en las cadenas.
— Como cadena se puede entender también ^L, que equivale a la pareja de órdenes: retorno de carro y nueva línea.

ED proporciona con la función M la capacidad de hacer que las órdenes especificadas se repitan más veces.

La forma es:

nmsecuencia-órdenes-de-ED

repite "n" veces la secuencia indicada.

Si "n" vale Ø ó 1 la secuencia se repetirá un número indefinido de veces, hasta que un mensaje de error señale la imposibilidad de una nueva ejecución.

Hay que notar que en la secuencia no es posible introducir otra orden M. Por ejemplo, operando sobre el fichero SERVICIO.LIB compuesto por:

1: Los dos obsequios fueron
2: ofrecidos a las dos personas
3: intervenidas

la orden:

:*mbsdos ^Zcuatro ^Zb#t

producirá:

1: Los cuatro obsequios fueron
2: ofrecidos a las dos personas
3: intervenidas.
1: Los cuatro obsequios fueron
2: ofrecidos a las cuatro personas
3: intervenidas

BREAK #AT^Z

1:*-

Si el usuario crea un fichero con extensión LIB (por ejemplo, SERVICIO.LIB) podrá explotar la orden

Rnombre-del-fichero ^Z (en este caso RSERVICIO)

para introducir todo lo que está incluido en el fichero especificado después del puntero en el buffer.

Esta función permite introducir un bloque de texto en distintos puntos del buffer, desplazando el puntero donde sea necesario.

Volvamos al fichero EDICION.TXT y pulsemos:

:*#a
1:*-b

:*rservicio
:*b#t

1: Nos preparamos a mostrar un fichero
2: creado por ED.
3: Los cuatro obsequios fueron
4: ofrecidos a las cuatro personas
5: que han intervenido
1:*e

(guardamos esta última versión).

En cambio, si el usuario quiere desplazar de posición un bloque de "n" líneas podrá utilizar la orden

nX

que transporta "n" líneas a partir del puntero a un fichero llamado

X$$$$$$.LIB

y, después de haber desplazado el puntero donde hace falta, el usuario podrá pulsar

R

e introducir las líneas del fichero creado de esta forma en el buffer.

La orden ØX limpiará el fichero.LIB.

En el ejemplo citado, partiendo de la ultima versión, sería:

:*#A
1:*2:x
2:*-br
:*Øx
:*3:x
3:*6:r
7:*2:2k
2:*b#t
1: Nos preparamos a mostrar un fichero
2: ofrecidos a las cuatro personas
3: que han intervenido.
4: Los cuatro obsequios fueron
5: creado por ED
1:*-

Utilizaremos ahora un artificio sobre la manera de obrar de ED.

Se envía una orden de movimiento, como E o H, con lo cual todos los ficheros temporales se destruirán. Cuando se quiere guardar el contenido de uno de ellos se envía ^C, que provocará el fin de la sesión de conexión pero sin destruir ningún fichero. Por lo tanto, resultará fácil modificar con REN la extensión de los ficheros.

Los mensajes de ED

En el momento que ED se encuentra con que no puede efectuar una orden, señala al usuario su dificultad de cuatro maneras distintas, en las cuales C indica el carácter respecto del cual se ha verificado el error.

1. BREAK? AT c

cuando ED no ha reconocido ninguna orden que se puede efectuar.

2. BREAK>AT c

cuando el fichero a corregir supera el espacio disponible en el buffer o si se especifican más de los 100 caracteres permitidos para las funciones que manejan cadenas.

Es, de todos modos, un error recuperable por medio de una orden de eliminación de caracteres, de líneas o de traslado desde el buffer al fichero temporal.

3. BREAK # AT c

cuando no se puede ejecutar la orden por el número de veces requerida.

4. BREAK O AT c

cuando es imposible actuar en el fichero.LIB con la orden R.

Si el error le obliga a relanzar el CP/M, acuérdese que existe una copia del fichero (con extensión BAK) que permite volver a utilizar la versión original del fichero origen.

De hecho, es suficiente, previa comprobación de la presencia de los ficheros, enviar la secuencia

```
A>ERA nombre-fichero.ext
A>REN nombre-fichero.ext=nombre-fichero.BAK
```

ED también controla todas las demás situaciones de transferencia que se encuentren en el CP/M.

CAPITULO VII

EDITANDO FICHEROS: EL COMANDO DDT

Tanto éste como los dos capítulos siguientes, contienen algunas referencias a aspectos bastante técnicos que precisan para su mejor comprensión un cierto conocimiento previo de la arquitectura e instrucciones de los microprocesadores 8080 y Z80. Aunque son temas muy puntuales recomendamos a todos aquellos lectores que deseen obtener el mayor provecho posible que acudan a los manuales y bibliografía disponible sobre dichos microprocesadores. (Nota del coordinador.)

Modificación interactiva de programas

En el mundo de la Informática, el "bug" (chinche) simboliza el error que se esconde entre las instrucciones del programa, un error muy difícil de identificar que sólo se logra descubrir fijándose bien en lo que puede "contaminar" la buena calidad del programa.

Así, parece muy apropiado llamar DDT (Dynamic Debugging Tool = Instrumento para la corrección interactiva) al conjunto de técnicas usadas en la corrección de los programas.

Al igual que el ED es necesario para la corrección de los ficheros de texto, el DDT representa el nuevo instrumento que nos permite actuar directamente en los ficheros binarios.

Como se verá más tarde, el DDT permite realmente modificar toda clase de ficheros, ya que actúa sobre los datos procesados por el ordenador.

El DDT, como el ED, es algo más que una sencilla orden; una vez llamada se instala en una zona auxiliar de la memoria para permitir la utilización de órdenes particulares que, de otra manera el CP/M no reconocería.

Puede ser llamado sin otra especificación en el caso de no actuar en un fichero, sino sobre el contenido de la memoria:

```
A>DDT
—
```

Su prompt es representado por un guión.

Si, en cambio, se indica el nombre del fichero objeto de las modificaciones:

```
A>DDT nombre-fichero.ext.
DDT vers zz,zz
NEXT    PC
xxxx    yyyy
```

carga el fichero especificado en la memoria central a partir de la posición 100H.

zz,zz representa la versión instalada del DDT.

xxxx indica la dirección inmediatamente sucesiva a la última posición ocupada por el fichero (por lo tanto xxxx H-100H-1H es el número efectivo de posiciones ocupadas por el fichero).

yyyy es la dirección de la primera instrucción que se ejecutará en el momento del lanzamiento del programa.

Por lo general, el DDT se utiliza para modificar los ficheros de formato.HEX que derivan del proceso de traducción de las instrucciones de un programa fuente a los códigos correspondientes (acuérdese de la utilización de ASM) y también en formato.COM, es decir, en la manera en que se presenta un programa que ya se puede ejecutar.

La diferencia entre estos dos ficheros consiste en los distintos valores contenidos en el registro PC en el momento de su carga.

Los ficheros.COM, efectivamente, representan una secuencia ordenada de sentencias que se pueden ejecutar; por lo tanto, el PC tendrá el valor 100H; es decir: la posición en que se encuentra la primera orden que se ejecutará es también la primera del área de memoria en que se ha cargado el fichero.

Los ficheros.HEX, sin embargo, requieren aún un proceso de absolutización que los vuelva ejecutables; por tanto, el valor del PC debe de ser directamente buscado en los registros del fichero.

Carga del fichero (input, read)

La manera más utilizada para cargar un fichero a corregir es especificándolo en el momento de llevar a cabo la orden DDT.

```
A>DDT nombre-del-fichero.ext
```

Como esta carga se efectúa en dos fases distintas, se pueden enviar las órdenes relativas separadamente para obtener un efecto distinto.

```
A>DDT
-Inombre-del-fichero.ext
-R
NEXT    PC
xxxx    yyyy
```

el comando I graba el nombre del fichero objeto de modificación en un área especial de la memoria, mientras R realiza la carga efectiva del contenido del fichero en la memoria central.

Esta secuencia distinta permite enviar varias veces el comando R sin necesidad de modificar el nombre del fichero grabado.

La forma Rn permite añadir el valor "n" a cada dirección especificada en el comando.

Las instrucciones List y Assembly

Los ficheros binarios son el resultado de una operación de conversión de las instrucciones-fuente a las equivalentes formas hexadecimales.

Muchas veces es necesario recorrer el camino contrario, o sea conocer (cuando es posible) la secuencia originaria de instrucciones que corresponden a los códigos hexadecimales que componen un fichero, es decir: desensamblar el fichero.

El comando

```
-L134,13B
```

visualizará las instrucciones correspondientes al contenido de las posiciones de memoria comprendidas entre 134 y 13B, con las oportunas señalaciones en los casos que no se correspondan con instrucción alguna.

El comando L controla un puntero de las posiciones visualizadas, lo que permite el envío de varios comandos L en secuencia.

Nótese que muchas instrucciones del Ensamblador ocupan más de una posición de memoria: por tanto, 12 bytes difícilmente corresponden a 12 mensajes.

En caso de que se especifique una dirección detrás de L, el puntero se desplazará sobre esa posición antes de desensamblar las instrucciones. Es decir:

-L2CF

mostrará las instrucciones contenidas a partir de la dirección 2CF.

Cuando se desee efectuar correcciones a nivel de fichero origen es posible, mediante el comando A, introducir las instrucciones deseadas, y el fichero será modificado directamente en la forma hexadecimal correspondiente a la instrucción introducida (o sea ensamblada).

Imaginemos que queremos modificar el contenido de algunas posiciones introduciendo los códigos correspondientes a las instrucciones

MOV C,4H y JNZ START

en posiciones a partir de 12C. La secuencia operativa será:

```
-A12C
-MOV C,4H
-JNZ START
-<RETURN>
```

en cuanto pulsemos la tecla RETURN damos fin a la inserción de la secuencia de instrucciones Assembler.

Es necesario resaltar que la sección de ensamblaje-desensamblaje del DDT se instala en las direcciones más altas de la memoria destinada a acoger los ficheros del usuario. Cuando el fichero a corregir es de dimensiones apreciables puede suceder que abarque la parte de memoria ocupada por estos comandos, haciéndoles inutilizables.

En este caso el sistema responderá con

?

a cada requerimiento de A o de L.

Manejarse en hexadecimal: comandos Display, Fill, Move y Set

Para examinar el contenido de las posiciones de memoria comprendidas entre dos direcciones, por ejemplo 100H y 140H, se utiliza el comando D (Display) en la forma

-D100, 140

El formato de la respuesta será parecido al que se mostró con el comando DUMP.

De hecho, el contenido de la memoria será mostrado en líneas de 16 caracteres (cada posición contiene un byte, representado por dos cifras hexadecimales), y en la parte derecha de la línea se visualizarán los correspondientes símbolos de la tabla ASCII.

También en este caso si la cifra hexadecimal contenida en una posición no se corresponde a ningún símbolo imprimible aparecerá en su lugar un punto.

El comando

-D13C

visualizará 16 líneas, de 16 caracteres cada una, a partir de la posición 13C.

La forma

-D

visualizará 16 líneas a partir de la posición a la que haga referencia el puntero que, también en este caso, avanza al mismo tiempo que los mensajes visualizados, lo que permite sucesivas utilizaciones de D sin especificar número de dirección alguna.

Nótese que la dirección inicial del grupo de 16 líneas es siempre múltiplo de 16; si la posición de principio dada no corresponde a una dirección múltiplo de 16, la primera línea visualizada no estará compuesta por 16 valores, sino sólo por los suficientes para hacer salir la línea sucesiva con un múltiplo de 16.

Así que, en caso de posiciones que contienen un texto

-D108,12F

la respuesta será:

```
0108  48 52 20 61 71 75 69  He aquí
0110  65 6A 65 6D 70 6C 6F 20 64 65 20 74 65 78 74 65  ejemplo de texto
0120  20 6D 6F 64 65 6C 6F 2E 0D 0A 1A 00 00 00 00 00  modelo.
```

Cuando se tiene la necesidad de llenar una zona entera de la memoria con el mismo valor se puede utilizar el comando F (Fill).

Si, por ejemplo, se quieren llenar con espacios blancos (20H) todas las posiciones entre 13CF y 1B05 se hará:

-F13CF, 1B05, 20

un comando D sobre el mismo grupo de posiciones permitirá controlar la exactitud de la ejecución.

Otra necesidad que se puede satisfacer de manera rápida es la referente a la copia de un grupo de posiciones en otra zona de la memoria.

El comando M (Move) permite especificar la dirección de la primera y de la última posiciones que delimitan el sector a duplicar y la dirección inicial del área que deberá acoger estos valores.

Por tanto, el comando

```
-M1300, 1360, 8000
```

duplicará el contenido de las posiciones cuya dirección va de 1300H a 1360H en el área de memoria que parte de 8000H.

La modificación del contenido de una posición de memoria es posible sólo desplazando un puntero de referencia sobre ella.

El comando que realiza este desplazamiento es S (Set).

```
-S35B
```

permitirá visualizar el contenido de la posición 35B y, en el mismo momento, permitirá al DDT darle un valor nuevo:

```
-S35B
035B 65-
```

indica que el contenido actual es 65H (letra "e"). Si el usuario introduce una cantidad numérica nueva, ésta será vista como el nuevo contenido a asignar a la posición. Si, en cambio, pulsa RETURN el valor actual será conservado y en pantalla se volverá a presentar la opción para la posición inmediatamente siguiente:

```
035C 41-
```

Este mecanismo permite la modificación del contenido de las posiciones una tras otra hasta que el usuario, para terminar, introduzca un punto un valor no permitido (un valor numérico fuera del campo 0-FF).

Supongamos querer introducir la palabra "cine" en las direcciones a partir de la 7000H, cuyo contenido actual sea "digo".

Se obtendrá la secuencia

```
-S7000
7000 64 43
7001 69                    (sólo RETURN)
7002 67 6E
7003 6F 65
7004 20
```


Controle con D el actual contenido de la memoria después de cada una de estas correcciones.

Este método puede ser utilizado, por ejemplo, para crear una versión personalizada del CP/M con los mensajes del S.O. y de los comandos no residentes, donde sea posible, en español. Es suficiente utilizar la primera parte de SYSGEN, lanzar DDT, buscar las posiciones en las que están contenidas las frases estándar usadas por el CP/M y... "traducirlas". Después se usa la segunda parte de SYSGEN.

Para los ficheros.COM debemos usar, en cambio, DDT y SAVE sólo.

Control de la ejecución: Go, Examine, Untrace, Trace

El DDT permite también la corrección interactiva de un programa siguiendo paso tras paso todo su funcionamiento.

El comando G (Go)

```
-G
```

permitirá la ejecución del programa a partir del valor contenido en el PC, el cual, como se recordará, conserva la dirección de la posición de memoria en donde está contenida la primera instrucción a ejecutar.

La forma

```
-Gn
```

permite hacer partir la ejecución de la instrucción contenida en la dirección "n".

Por la estructura de la memoria, el comando

```
-GØ
```

provoca el final de la sesión de DDT y la vuelta al CP/M, sin que se altere el contenido de la memoria, ya que salta a la posición ØH y ejecuta la reentrada del CP/M.

Si el programa no está estructurado de manera que vuelva al DDT, el control de su ejecución será imposible. Para impedir esto se utiliza la forma

```
-G,n
```

El programa se parará entonces justo antes de ejecutar la instrucción contenida en la dirección "n" y devuelve el control de las operaciones al DDT.

Es intuitivo que la forma

-Gn1, n2

permitirá ejecutar los pasos contenidos entre la instrucción de la dirección n1 y la instrucción de la dirección n2.

De todas maneras, como la ejecución de un programa no es prácticamente nunca lineal por las diversas instrucciones de salto condicional e incondicional, es necesario poder bloquear en más de un sitio el flujo de instrucciones para reobtener el control del programa.

En este caso la expresión será:

-Gn, s1, s2

el programa, habiendo pasado por la instrucción contenida en la dirección "n", se para en cuanto encuentra la instrucción de S1 ó S2.

Al comando G le sigue normalmente el comando X (eXamine), el cual permite examinar el contenido actual de los registros del microprocesador.

-X

producirá una línea de información organizada así:

CxZxMxExIx A=yy B=yyyy D=yyyy H=yyyy S=hhhh P=hhhh aaa

en donde:

- x representa el valor (Ø/1) del bit de los indicadores C, Z, M, E e I
- cada pareja de "y" representa el byte contenido en los registros A, BC, DE, HL, SP y PC
- aaa especifica la instrucción Assembler correspondiente al contenido de la posición actual

Cuando se quiere modificar el valor de un registro se utiliza el comando Xr; el mecanismo de modificación será el mismo que el comando S.

Imaginemos que queremos modificar el Program Counter del valor actual (ØH) a 1ØØH.

-XP
P=ØØØØ 1ØØ

se podrá verificar, con X, la corrección efectiva.

El comando U (Untrace) permite observar, con el mismo formato que X, el contenido de los registros del microprocesador al final de un número de pasos deseado.

Si, por ejemplo, las posiciones 12C a 13F contienen 10 pasos de programa, una vez que introducimos 12C en PC, las expresiones

-G, 13F
*Ø13F
-X

y la

-U1Ø

producirán la visualización de la misma información.

En cambio, cuando se desee controlar el contenido de los registros después de cada paso de programa, se deberá utilizar el comando T (Trace).

Si se quiere, por ejemplo, controlar lo que sucede durante la secuencia de las siguientes 7 instrucciones se deberá escribir

-T7

para obtener en pantalla 7 líneas consecutivas parecidas en la forma, pero, obviamente, con sus diferentes valores, a la producida por el comando X.

De todas maneras, hay que subrayar que también X, U y T están contenidos en una sección que se instala en la parte alta de la memoria y que, como A y L, pueden ser borrados si se carga un fichero muy largo.

Conservar las correcciones

La conclusión natural de una sesión de edición de ficheros es la grabación sobre el disco de la versión modificada del fichero. Esto se obtiene gracias al ya conocido comando SAVE.

Es oportuno notar que la mínima unidad controlable por SAVE es la página de memoria, o sea 256 bytes.

De hecho, SAVE graba también posiciones no deseadas en número suficiente como para completar esas 256 (100H) y, como una vez llamado el DDT, NEXT visualiza la dirección de la posición inmediata a la última ocupada por el fichero, un cálculo equivocado de la dirección provocará un derroche aún mayor del espacio ocupado por el fichero.

Ya se ha propuesto, como método para calcular el número de páginas a indicar en el comando, ignorar las dos últimas cifras de NEXT.

Recordemos también que NEXT representa la dirección siguiente a la última ocupada por el fichero; por lo tanto, antes de ejecutar la transformación hexadecimal-decimal es necesario restar 1H a la cantidad visualizada. Esto significa, en el caso en que las últimas dos cifras sean ØØ (por ejemplo CØØ), que el número efectivo que representa la dimensión del fichero termina con FF (o sea, BFF).

Si no siguiéramos este camino se salvaría una página más de lo necesario.

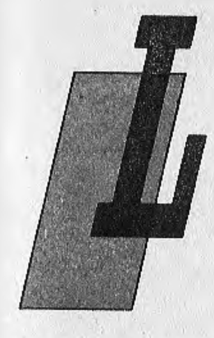
De hecho, en el ejemplo propuesto

BBF producirá SAVE 11 nombre-del-fichero.ext
porque B (o mejor BFF-FF) equivale a 11, mientras que
CØØ producirá SAVE 12 nombre-del-fichero.ext
en cuanto C(CØØ-ØØ) equivale a 12.

CAPITULO VIII

BAJO LA PIEL DEL CP/M

Los módulos-base del CP/M



La estructura general del CP/M se resume en el esquema de la figura 1.

El CP/M opera dividiendo la memoria central del ordenador en las cuatro secciones que muestra la figura 2 y que son:

La página cero	De la posición ØH —la H indica Hexadecimal— a la FFH (256 bytes) están contenidos unos parámetros fijos a los que el sistema hace referencia para el control de las interconexiones entre los módulos que componen el CP/M.
TPA	Es el área en que son cargados los programas a ejecutar (Transient Program Area) y su dimensión varía según la versión instalada. Parte de 1ØØH (TBASE) y se extiende hasta CBASE.
CCP	Se extiende de CBASE a FBASE. En esta zona están contenidos los comandos que interactúan con el usuario (Console Command Processor). El CCP es responsable del diálogo y de las comunicaciones usuario-S.O.
FDOS	Se extiende desde FBASE hasta el final de la memoria. Contiene las subrutinas que el ordenador necesita para ejecutar las tareas requeridas (File

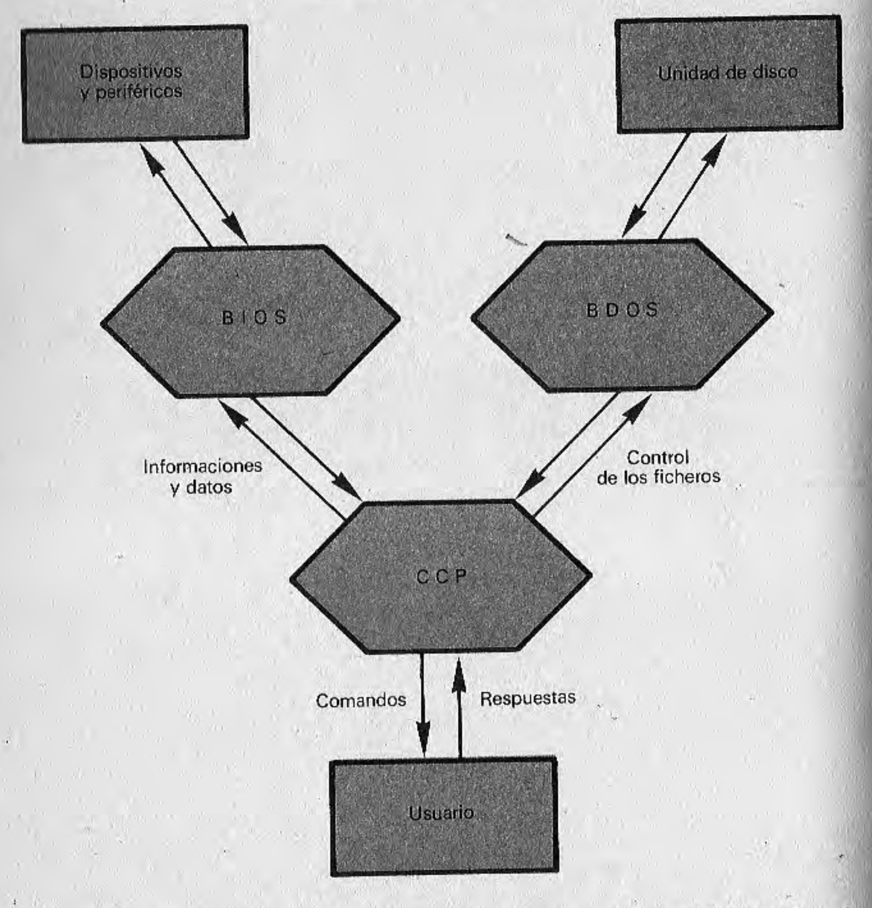


Figura 1.— Estructura general del sistema operativo CP/M. La figura muestra el papel desempeñado por los módulos fundamentales en la comunicación del ordenador con el usuario y los periféricos, divididos a su vez en la unidad de discos y otros.

Disk Operating System). Está compuesto por dos secciones diferentes:

- BIOS (Basic Imput Output System). Constituye el conjunto de las subrutinas que realizan la conexión entre las funciones del CP/M y las características físicas del ordenador; al servicio

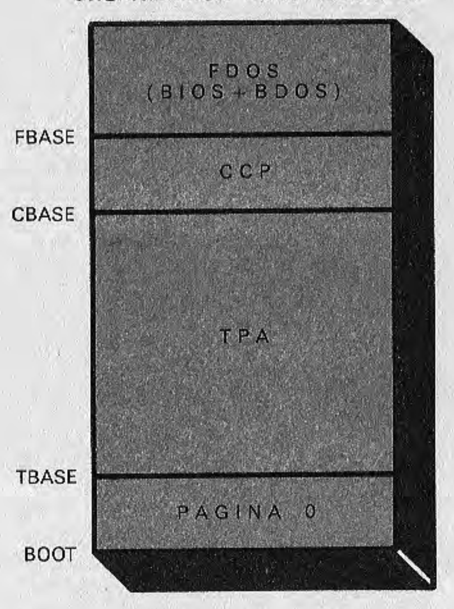


Figura 2.— Mapa de memoria del CP/M, dividido en cuatro partes que corresponden a cuatro módulos de control más la página 0.

del CCP controlan los periféricos enlazados (excepto la unidad de discos).

- BDOS (Basic Disk Operating System). Constituye el conjunto de las subrutinas que se ocupan por completo del control de la unidad de discos.

Cuando aparece el aviso A> el CCP es el módulo activo. Examina todo lo que el usuario teclea y ejecuta las órdenes requeridas. Si una orden no es reconocida (por errores de sintaxis o por inexistencia del fichero.COM), el CCP reescribe los caracteres pulsados seguidos de un punto interrogativo.

En caso de tratarse de un comando no residente, el correspondiente fichero es cargado en la TPA (área de memoria destinada a este fin) y ejecutado.

Si la orden dada por el usuario especifica algún fichero, el CCP se sirve del FCB (File Control Block, bloque de control de fichero); éste consiste en una secuencia de 36 caracteres, contenidos en la página Ø, que permite grabar el modo en que los fi-

cheros son controlados, para lo cual requiere la intervención del BDOS.

El CCP y el BDOS son módulos independientes de la estructura física del ordenador, ya que controlan sólo unos parámetros y se sirven, en cuanto al hard, del BIOS.

El BIOS es la famosa sección del CP/M que puede ser modificada para adaptarla a cualquier configuración hard; a petición del CCP informa sobre el estado del periférico que interese mediante unos valores que evitan al módulo que desea conocerlo tener que saber cómo reacciona el periférico o la manera en que su estado puede ser investigado.

El BIOS representa la estructura que lleva consigo todas las informaciones que los módulos del CP/M se intercambian entre ellos.

En definitiva, el mecanismo de ejecución de una orden es el siguiente:

— el CCP recibe una orden;
— si no es correcta la rechaza;
— si es un comando residente lo ejecuta;
— si es un comando no residente carga el fichero.COM en la TPA;
— requiere la intervención del BDOS para el control de los ficheros;
— requiere la intervención del BIOS para efectuar las oportunas conexiones con los periféricos.

El BIOS

El BIOS se compone de una serie de subrutinas que pueden ser requeridas por varias órdenes del CCP y que proporcionan como respuesta unos valores relacionados con el estado de los periféricos asociados.

Estas subrutinas utilizan unas tablas que "describen" la tipología característica del hardware de ese ordenador. Por tanto, con sólo modificar las tablas podremos obtener una nueva versión del CP/M, apta para otro diseño hardware.

Para acceder a las subrutinas del BIOS se utiliza la localización 5H de la página Ø; debemos proporcionar un número que representa el código de la función requerida y las informaciones necesarias, que las subrutinas utilizarán como datos de partida.

En general, el número de función se debe poner en el registro C, y la dirección en la que se busca la información, en el registro (doble) DE.

Los datos que las funciones devuelven pueden ser recuperados en el registro A si se trata de un byte, y en el registro doble HL si la respuesta consiste en una pareja de bytes.

Examinemos ahora las características más importantes de las subrutinas, recordando que un tratamiento más amplio, adecuado para quienes escriban programas con accesos directos a estas funciones, se puede encontrar en la documentación original de la Digital Research.

Cod.	Valor en C (hexadec)	Objetivo de la función	Datos necesarios	Datos proporcionados
Ø	Ø	Puesta en marcha del sistema.	—	—
1	Ø1	Lectura de datos desde la consola. Lee desde CON:	—	A=Car.ASCII
2	Ø2	Escritura de datos sobre la consola. Escribe sobre CON:	E=Car.ASCII	—
3	Ø3	Lectura desde el lector. Lee desde RDR:	—	A=Car.ASCII
4	Ø4	Escritura sobre la perforadora. Escribe sobre PUN:	E=Car.ASCII	—
5	Ø5	Escritura sobre la impresora. Escribe sobre LST:	E=Car.ASCII	—
6	Ø6	Operaciones de I/O sobre la consola (lee un carácter y el estado de la consola o envía un carácter).	E=FFH E=FFH E=Car.ASCII	A=Car.ASCII A=Estado.
7	Ø7	Lee el estado de I/O.	—	A=byte I/O
8	Ø8	Asigna el estado de I/O.	E=byte I/O	—
9	Ø9	Imprime un buffer desde una dirección = hasta un $	DE=Dirección	—
1Ø	ØA	Llena un buffer (a partir de la dirección) con caracteres desde CON:	DE=Dirección	Buffer lleno
11	ØB	Lee el estado de CON:; si está listo un carácter devuelve FFH, si no ØH.	—	A=FFH/ØH

La página 0, la TPA y el CCP

En las direcciones de la página 0 están contenidas funciones que realizan tareas específicas. Veamos las direcciones de mayor interés.

— En la localización 0H reside la función que permite la puesta en marcha del sistema, ya que efectúa un salto a la localización del BIOS correspondiente. Por lo tanto, un programa que contenga un salto a la dirección 0H, permitirá esta operación del CP/M.

— La localización 4H contiene la indicación de la unidad de disco por defecto. Contiene un salto al BDOS y constituye el medio a través del cual se solicita directamente a éste la información deseada.

— La dirección 5CH (y siguientes hasta un total de 36 direcciones) contiene el FCB que usa el CCP.

— La localización 80H (y siguientes hasta un total de 128 posiciones) constituye el área de DMA (Direct Memory Access) por defecto, o sea, el buffer de tránsito para los movimientos de datos disco-memoria.

El CCP se carga inmediatamente por encima del área destinada a conservar los programas a ejecutar (TPA), pero, una vez en marcha un programa, su tarea está acabada.

El CP/M permite que un programa de grandes dimensiones ocupe, además de la TPA, parte del área del CCP, por lo que es posible introducir en un programa un salto a la localización BOOT (en general 0H) que realiza la inicialización del sistema y permite al usuario dialogar de nuevo con el CCP.

El BDOS

La forma en la cual las informaciones se graban sobre el disco requiere un eficaz control de los ficheros, ya que cada disco está dividido en pistas concéntricas, subdivididas a su vez en sectores contiguos.

El acceso a los ficheros en tiempos cortos y la presencia simultánea de muchos ficheros, que se disputan el espacio libre, impiden que los datos contenidos en un fichero estén, en realidad, concentrados en la misma zona del disco. Por lo tanto, es necesario que a cada fichero se le asocie un mapa que determine la localización física de sus datos sobre el disco.

Cada fichero grabado en el disco dispone, pues, de una parte informativa, llamada FCB (File Control Block), que se conserva en las primeras pistas del disco, inmediatamente después de las destinadas al S.O.

Cuando se necesita acceder a un fichero se "abre" en la TPA el espacio necesario para trasladar el correspondiente FCB. En la primera parte de la TPA se graba el FCB, mientras la segunda es controlada directamente por el CP/M, que se sirve de ella para grabar las informaciones necesarias en las operaciones de acceso.

Al final de todas las operaciones la versión actualizada del FCB, que está en la TPA, es duplicado sobre el disco y representará desde ese momento el mapa de partida válido para nuevos accesos.

Veamos ahora el significado de los 36 bytes que componen un FCB (File Control Block):

Nº orden	Nº bytes	Significado
1	1	Código de la unidad (A-P).
2-9	8	Nombre del fichero (en letras mayúsculas).
10-12	3	Extensión del fichero (en letras mayúsculas): Si el primer bit del 10 es 1, el fichero es de tipo R/O; si el primer bit del 11 es 1, el fichero es de tipo SYS.
13	1	Valor de Ext; actualizado durante el acceso al fichero.
14	1	Reservado para uso interno del sistema.
15	1	Reservado para uso interno del sistema. Se pone a 0 en las subrutinas de apertura de ficheros.
16	1	Número de registros del actual extent (va de 0 a 127).
17-32	16	Utilizados por el CP/M para el control interno.
33	1	Número del registro a controlar en la siguiente operación de escritura o lectura.
34-35	2	Número de registros en caso de acceso directo (de 0 a 65.535).
36	1	Señal de overflow en caso de acceso directo (o aleatorio).

Cuando un programador controla el FCB establece, de hecho, el valor de casi todos los primeros 16 bytes y del 33. Los otros dependen del programa en concreto.

El CCP utiliza una zona particular de la página Ø, a partir de la localización 5CH, como FCB de los ficheros especificados en las órdenes.

También el BDOS está constituido de subrutinas, a las que se accede a través de la localización 5H de la página 0, debiendo proporcionar unos particulares códigos de función en el registro C y otras eventuales informaciones necesarias.

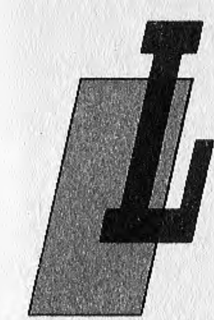
Veamos las características sobresalientes de estas subrutinas:

Cod.	Valor en C (hexadec.)	Objetivo de la función	Datos necesarios	Datos proporcionados
12	ØC	Número de la versión	—	HL=Número versión.
13	ØD	Inicialización del Disco del Sistema (permite el cambio de los discos).	—	—
14	ØE	Selección de un disco.	E=Núm.Disco.	—
15	ØF	Apertura de un fichero.	DE=Dirección del FCB.	A=Código del directorio.
16	1Ø	Cierre de un fichero.	DE=Dirección del FCB.	A=Código del directorio.
17	11	Búsqueda de un fichero.	DE=Dirección del FCB.	A=Código del directorio.
18	12	Búsqueda de la siguiente ocurrencia de un fichero.	—	A=Código del directorio.
19	13	Borrado de un fichero.	DE=Dirección del FCB.	A=Código del directorio.
2Ø	14	Lectura secuencial de un fichero.	DE=Dirección del FCB.	A=Código de error. DMA=Registro.
21	15	Escritura secuencial en un fichero.	DE=Dirección del FCB.	A=Código de error. DMA=Registro.
22	16	Creación de un fichero.	DE=Dirección del FCB.	A=Código de error.
23	17	Cambio de nombre de un fichero.	DE=Dirección del FCB.	A=Código de error.
24	18	Vector de las unidades de disco activos.	—	HL=Vector.
25	19	Unidad seleccionada.	—	A=Unidad sel.
26	1A	Modificación del DMA.	DE=Nuevo DMA.	—
27	1B	Vector de posición del espacio ocupado.	—	HL=Dirección vector.

Cod.	Valor en C (hexadec.)	Objeto de la función	Datos necesarios	Datos proporcionados
28	1C	Protección del disco selec. contra la escritura.	—	—
29	1D	Vector de los discos protegidos contra la escritura.	—	HL=Vector.
3Ø	1E	Modificación de los atributos de un fichero.	DE=Dirección del FCB.	—
31	1F	Parámetros del disco.	—	HL=Dirección del BPD.
32	2Ø	Control del Código de Usuario.	E=FFH E=Nuevo Cód. Usuario.	A=Cód. Usuario. —
33	21	Lectura en un fichero de acceso directo.	DE=Dirección del FCB.	A=Cód. de error. DMA=Registro.
34	22	Escritura en un fichero de acceso directo.	DE=Dirección del FCB.	A=Cod. de error. DMA=Registro.
35	23	Cálculo de la dimensión virtual de un fichero.	DE=Dirección del FCB.	Bytes 34-36 del FCB=Dirección registro que sigue a EOF (final del fichero).
36	24	Número del registro en un fichero de acceso secuencial.	DE=Dirección=del FCB.	Bytes 34-36 del FCB.=Posición.
37	25	Inicialización de las unidades.	DE=Vector de las unidades.	A=OOH.
4Ø	28	Escritura en un fichero de acceso directo (llena el registro con ceros antes de escribir).	DE=Dirección del FCB.	A=Código de error. DMA=registro.

CAPITULO IX

DOS EJEMPLOS EN ENSAMBLADOR



La mejor manera de llegar a conocer el significado de los comandos de un S.O. y de los matices proporcionados por sus diversas opciones es practicar sobre un caso real.

La escritura de un fichero de texto (con ED), su conversión (con ASM) y el control del funcionamiento del programa resultante, representan ciertamente el mejor banco de pruebas y el mejor ejercicio práctico que se puede proponer. Vamos a examinar, pues, dos ejemplos distintos de estos pequeños programas.

Búsqueda del máximo entre varios números

El primer ejemplo consiste en encontrar el máximo entre varios números y colocarlo en una situación específica.

El listado en Ensamblador es:

```
        ORG   100H     ;INICIO DE LA TPA
        MVI   B,NTOT   ;LONGITUD DEL VECTOR DE LOS DATOS.
        MVI   C,0      ;MAXIMO=0
        LXI   H,LISTA  ;PRIMER VALOR DEL VECTOR.
LOOP:   MOV   A,M      ;LEE EL NUMERO
        CMP   C        ;ES MAYOR QUE EL NUMERO EN C?
        JC    NOMAY    ;VA A NOMAY SI NO LO ES.
        MOV   C,A      ;SI LO ES GUARDA EN C EL NUEVO MAXIMO.
NOMAY:  INX   H        ;LEE EL SIGUIENTE VALOR.
        DCR   B        ;ESTA ACABADO EL VECTOR?
        JNZ   LOOP     ;VA A LOOP SI NO LO ESTA
        MOV   A,C      ;COPIA EN A EL MAXIMO ENCONTRADO.
```

```
         STA    MAXVAL      ;LO PASA A LA DIRECCION MAXVAL.
         RST    7           ;VUELVE AL DDT.
                            ;AREA DATOS
LISTA:   DB     5,2,3,9, 1,2,0,8
NTOT:    EQU    $-LISTA     ;LONGITUD DEL VECTOR
MAXVAL:  DS     1           ;DIRECCION DONDE DEJARA EL MAXIMO
         END
```

Para realizar el ejercicio tendremos que crear un fichero con ED:

A>ED MAXIMO.ASM ↵

y así, abandonado ED, convertirlo:

A>ASM MAXIMO ↵

y verificar si todo es correcto con:

A>TYPE MAXIMO.PRN ↵

Una vez comprobado que la redacción es correcta, se podrá proceder a la ejecución:

A>DDT MAXIMO.HEX ↵

la respuesta será:

```
DDT vers.xxxx
NEXT PC
O11F 0000
```

Para efectuar el programa se tendrá que poner el PC a 100H:

```
-XP
P=0000 100
```

y ejecutarlo

```
-G
*0116 (dirección última instrucción)
```

al final, mediante:

-D117,11F

se verán los 8 valores de los datos utilizados (LISTA) y el valor máximo, depositado en la posición 11F.

Si el funcionamiento no es del todo exacto, deberemos seguir el proceso del programa con T para descubrir los eventuales errores.

Si modifica los valores utilizados como datos y su número, tenga presente que, como consecuencia de esto, cambiará también la dirección de la posición en la que se debe buscar el máximo después de la ejecución.

Una subrutina de ordenación

Se entiende por "sort" una operación de ordenamiento, en sentido decreciente o creciente, de una serie de valores.

Ejercitémonos sobre una subrutina que efectúa un ordenamiento en sentido creciente de una serie de 10 números.

El listado en Ensamblador es:

```
         ORG    100H        ;INICIO TPA
ORDENA:  LXI    H,INTER     ;DIRECCION DEL INDICADOR
         MVI    M,0         ;PRIMERA VUELTA DE EJECUCION
         LXI    H,I         ;DIRECCION DEL CONTADOR
         MVI    M,0         ;CONTADOR=0
CONFR:   MOV    A,M         ;A=CONTADOR (A=I)
         CPI    NTOT-1      ;SI I<(NTOT-1) MODIFICA C
         JC     CONT        ;CONTINUA SI I<=(NTOT-2)
         LXI    H,INTER     ;CONTROLA SI INDICADOR=0
         MOV    A,M
         ORA    A
         JNZ    ORDENA      ;TERMINA SI INDICADOR=0
         RST    7           ;VUELVE A DDT
CONT:    MOV    E,A         ;CARGA LISTA (I)
         MVI    D,0
         LXI    H,LISTA
         DAD    D
         DAD    D
         MOV    C,M         ;BYTE DE ORDEN BAJO EN A Y C
         MOV    A,C
         INX    H
         MOV    B,M         ;BYTE DE ORDEN ALTO EN B
         INX    H           ;DESPLAZA H & L A LISTA (I+1)
         SUB    M           ;CONFRONTACION CON LISTA (I)
         MOV    D,A
         MOV    A,B
         INX    H
         SBB    M
         JC     ENORD       ;SI ESTAN YA EN SECUENCIA
```

```
         ORA   D
         JZ    ENORD     ;SI SON IGUALES
         MOV   D,M       ;SI NO, EJECUTA EL INTERCAMBIO
         MOV   M,B
         DCX   H
         MOV   M,C
         DCX   H
         MOV   M,D
         DCX   H
         MOV   M,E
         LXI   H,INTER   ;INCREMENTA EL INDICADOR
         INR   M
ENORD:   LXI   H,I       ;INCREMENTA EL CONTADOR
         INR   M
         JMP   CONFR
                         ;AREA DATOS
INTER:   DB    0
I:       DS    1
LISTA:   DW    34,234,1,5 999,21,32 000,15 999; 20,16,30,
NTOT:    EQU   ($-LISTA)/2
         END
```

La secuencia que hay que seguir para lograr un correcto funcionamiento es la misma del ejercicio anterior, con la diferencia de que los datos numéricos, una vez transformados a hexadecimal, son representados cada uno por dos bytes y su tratamiento y lectura se lleva a cabo de manera inversa a la tradicional (primero el menos significativo —LSB— y luego el más significativo —MSB—). Así, el número 54FB (hexadecimal) será almacenado como FB y 54; FB representa el byte de orden menor (LSB) y 54 el byte de orden mayor (MSB).

Las órdenes serán:

```
A>ED      SORTNUM.ASM ↵
A>ASM     SORTNUM ↵
A>TYPE    SORTNUM.PRN ↵
A>DDT     SORTNUM.HEX ↵
DDT vers.xx.xx
NEXT      PC
015C      0000
-XP
P=0000    100
-G
*0118
```

y, al final de la ejecución:

```
-D148,15B
```


permitirá observar el contenido de las situaciones que van de 148 a 15B, en las que se hallarán los 10 valores iniciales en forma hexadecimal y en secuencia rigurosamente creciente, con el byte de orden menor grabado antes del byte de orden mayor (ØØ7D aparecerá como 7DØØ).

Como cada número, en este ejemplo, está compuesto por dos bytes, resultarán ocupadas 20 posiciones de memoria.

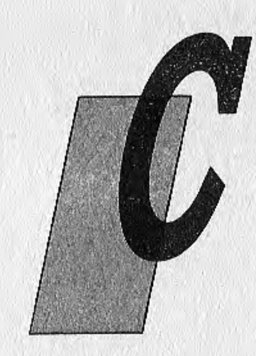
APENDICE A

TABLA DE EQUIVALENCIAS

BINARIO	DECIMAL	HEXADECIMAL	ASCII		BINARIO	DECIMAL	HEXADECIMAL	ASCII	BINARIO	DECIMAL	HEXADECIMAL	ASCII
0000000	0	00	NUL		0101011	43	2B	+	1010110	86	56	V
0000001	1	01	SOH	^A	0101100	44	2C	,	1010111	87	57	W
0000010	2	02	STX	^B	0101101	45	2D	-	1011000	88	58	X
0000011	3	03	ETX	^C	0101110	46	2E	.	1011001	89	59	Y
0000100	4	04	EOT	^D	0101111	47	2F	/	1011010	90	5A	Z
0000101	5	05	ENQ	^E	0110000	48	30	0	1011011	91	5B	[
0000110	6	06	ACK	^F	0110001	49	31	1	1011100	92	5C	
0000111	7	07	BEL	^G	0110010	50	32	2	1011101	93	5D	]
0001000	8	08	BS	^H	0110011	51	33	3	1011110	94	5E	^
0001001	9	09	HT	^I	0110100	52	34	4	1011111	95	5F	—
0001010	10	0A	LF	^J	0110101	53	35	5	1100000	96	60	'
0001011	11	0B	VT	^K	0110110	54	36	6	1100001	97	61	a
0001100	12	0C	FF	^L	0110111	55	37	7	1100010	98	62	b
0001101	13	0D	CR	^M	0111000	56	38	8	1100011	99	63	c
0001110	14	0E	SO	^N	0111001	57	39	9	1100100	100	64	d
0001111	15	0F	SI	^O	0111010	58	3A	:	1100101	101	65	e
0010000	16	10	DLE	^P	0111011	59	3B	;	1100110	102	66	f
0010001	17	11	DC1	^Q	0111100	60	3C	<	1100111	103	67	g
0010010	18	12	DC2	^R	0111101	61	3D	=	1101000	104	68	h
0010011	19	13	DC3	^S	0111110	62	3E	>	1101001	105	69	i
0010100	20	14	DC4	^T	0111111	63	3F	?	1101010	106	6A	j
0010101	21	15	NAK	^U	1000000	64	40		1101011	107	6B	k
0010110	22	16	SYN	^V	1000001	65	41	A	1101100	108	6C	l
0010111	23	17	ETB	^W	1000010	66	42	B	1101101	109	6D	m
0011000	24	18	CAN	^X	1000011	67	43	C	1101110	110	6E	n
0011001	25	19	EM	^Y	1000100	68	44	D	1101111	111	6F	o
0011010	26	1A	SUB	^Z	1000101	69	45	E	1110000	112	70	p
0011011	27	1B	ESC	^[	1000110	70	46	F	1110001	113	71	q
0011100	28	1C	FS	^	1000111	71	47	G	1110010	114	72	r
0011101	29	1D	GS	^]	1001000	72	48	H	1110011	115	73	s
0011110	30	1E	RS	^^	1001001	73	49	I	1110100	116	74	t
0011111	31	1F	US	^	1001010	74	4A	J	1110101	117	75	u
0100000	32	20	SPACE	—	1001011	75	4B	K	1110110	118	76	v
0100001	33	21	!		1001100	76	4C	L	1110111	119	77	w
0100010	34	22	"		1001101	77	4D	M	1111000	120	78	x
0100011	35	23	#		1001110	78	4E	N	1111001	121	79	y
0100100	36	24	$		1001111	79	4F	O	1111010	122	7A	z
0100101	37	25	%		1010000	80	50	P	1111011	123	7B	
0100110	38	26	&		1010001	81	51	Q	1111100	124	7C	|
0100111	39	27	'		1010010	82	52	R	1111101	125	7D	
0101000	40	28	(		1010011	83	53	S	1111110	126	7E	~
0101001	41	29	)		1010100	84	54	T	1111111	127	7F	DEL
0101010	42	2A	*		1010101	85	55	U				

APENDICE B

MENSAJES DE ERROR EN LA UNIDAD DE DISCOS



Cada comando del CP/M dispone de una casuística nutrida de los errores que pueden surgir durante su ejecución, pero existen también unos mensajes, independientes de la función con la que se está trabajando y que tienen que ver, principalmente, con el control de los discos (con d: se indica la unidad —drive— genérica):

Bdos Err On d: Bad Sector

Esta señalización es la que aparece cuando se produce una situación de error que tiene que ver con el hardware. Puede depender de un formateado equivocado del disco, del intento de acceder a partes de él estropeadas o, simplemente, de una equivocada puesta en marcha del sistema (ventanillas de la unidad abiertas, falta de alimentación, etc.). Es suficiente pulsar ^C para volver al CP/M o RETURN para hacer seguir el programa.

Bdos Err On d: File R/O

Señala el intento de escribir sobre un fichero protegido contra la escritura. Es necesario modificar su atributo a R/W por medio de STAT.

Bdos Err On d: R/O

Indica el intento de escribir sobre un disco contenido en una unidad designada como R/O o que en la unidad se ha sustituido el disco anterior por otro, sin haber enviado el oportuno ^C. La pulsación de cualquier tecla hace salir de nuevo el CP/M.

Bdos Err On d: Select

Es el mensaje que aparece cuando el usuario especifica una unidad inexistente. La pulsación de una tecla cualquiera hace salir al CP/M.

APENDICE C

CARACTERES DE CONTROL

BACKSPACE	Cancela el último carácter introducido y desplaza el cursor una posición hacia atrás.
CTRL-C	Arranca de nuevo el CP/M. Permite identificar el disco situado en la unidad de discos. Deberá usarse después de cambiarlo por otro nuevo y antes de usar éste.
CTRL-E	Genera un carácter de fin de línea, permitiendo la continuación de la orden en la línea siguiente.
CTRL-H	Equivale a Backspace.
CTRL-J	Provoca el fin de la introducción de datos al generar un RETURN.
CTRL-M	Idem (vuelta-carro).
CTRL-P	Envía a la impresora todos los caracteres que se presentan por pantalla. Funciona como flip-flop, por lo que una nueva pulsación desactivará esta función.
CTRL-S	Detiene la ejecución de una función que, pulsando una tecla cualquiera, volvería a activarse.
CTRL-U	Borra toda la línea.
CTRL-X	Idem.
CTRL-Z	Provoca el final de la introducción de datos en programas como PIP y ED.

APENDICE D

EXTENSIONES DE LOS FICHEROS

Algunas funciones son permitidas sólo si los ficheros a los que afectan presentan unas extensiones determinadas.

ASM — Representa la extensión que debe ser atribuida a un fichero cuando se le quiere someter al comando ASM. Debe contener, por tanto, el texto fuente de un programa escrito en Ensamblador.

BAK — Es la extensión atribuida a un fichero de texto cuando está sometido al comando ED. Sirve para disponer de una copia original del fichero que se está editando.

BAS — Es la extensión atribuida a un fichero creado por medio de los comandos del BASIC.

COM — Es la extensión que debe tener un fichero para que sea interpretado como un comando no residente. El comando LOAD crea un fichero parecido partiendo de un fichero (.HEX).

INT — Es la extensión atribuida al fichero generado por un compilador.

PRN — Es la extensión del fichero de información adjunto creado por el comando ASM.

SUB — Extensión de los ficheros creados con el CP/M, con comandos residentes o por programas. Para ser ejecutados en batch por medio de SUBMIT.

$$$ — Es la extensión atribuida a los ficheros temporales creados por comandos como ED y SUBMIT.

APENDICE E

INDICE ANALÍTICO DE COMANDOS

ASM	Transforma un fichero (.ASM) que contiene programas escritos en Ensamblador en otro fichero de extensión (.HEX) que contiene su correspondiente traducción a código hexadecimal. Así también crea un fichero (.PRN) que contiene información adicional (líneas fuente en Ensamblador con los mensajes de error).
DDT	Permite la modificación del contenido hexadecimal de los ficheros y el control del contenido de la memoria durante el funcionamiento de los programas.
DIR	Visualiza la lista de los ficheros contenidos en un disco.
DUMP	Visualiza el contenido de un fichero en caracteres hexadecimales.
ED	Permite el control de los ficheros de texto. Crea una copia de seguridad (.BAK) del fichero original.
ERA	Borra los ficheros especificados.
LOAD	Carga en memoria el fichero (.HEX) creando el correspondiente fichero (.COM).
MOVCPM	Genera una nueva versión del CP/M.
PIP	Permite el traslado de un fichero a cualquier periférico del sistema.
REN	Cambia el nombre del fichero especificado.
SAVE	Graba en disco con el nombre de fichero especificado el contenido de las páginas de memoria indicadas.

STAT	Controla las características y los parámetros de enlace de los periféricos.
SUBMIT	Ejecuta una serie de órdenes contenidas en un fichero (.SUB) y no introducidas por la consola.
SYSGEN	Traslada el CP/M de un disco a la memoria y de la memoria a otro disco.
TYPE	Visualiza el contenido del fichero especificado.
USER	Controla las áreas del usuario en las que están divididos los discos.
XSUB	Permite utilizar en los ficheros (.SUB) comandos que necesitan un buffer (como DDT y ED).

APENDICE F

PARÁMETROS DEL COMANDO PIP

B	Se utiliza para el traslado en bloques por un dispositivo de lectura continua. Los datos se ponen en un buffer que se vacía al recibir un carácter ASCII "X-OFF" (↑S). El tamaño del buffer depende de cada sistema.
Dn	Borra todas las columnas a partir de la n-ésima; se utiliza para dispositivos que no aceptan más de "n" columnas.
E	Visualiza el contenido de los ficheros trasladados.
F	Elimina los saltos de página de un fichero.
Gn	Se usa para copiar los ficheros contenidos en el área "n" del usuario.
H	Es obligatorio en el caso de traslado de ficheros que contengan datos en formato hexadecimal.
I	Lleva a cabo las mismas funciones que H en caso de ficheros en formato hexadecimal, pero elimina los registros nulos.
L	Cambia a minúsculas todas las letras mayúsculas.
N2	Antepone a cada línea un número de tres cifras (001; 002; etc.) y un carácter de tabulación.
N	Antepone a cada línea el número de secuencia y ":".
O	Transfiere ficheros objeto (no en código ASCII). Es obligatorio para concatenar ficheros binarios, en cuanto permite ignorar el carácter de fin de fichero.

Pn Incluye un salto de página cada "n" líneas. Si n=0 o n=1, el salto de página ocurre cada 60 líneas. Se verificará siempre un salto de página inicial.

Qss^Z El traslado terminará después de haber encontrado la cadena de caracteres "ss" que se especifique. El fin de cadena se marca con CTRL-Z.

R Es obligatorio cuando se quieren trasladar ficheros $ SYS.

Sss^Z El traslado empezará nada más encontrar la cadena de caracteres "ss" que se especifique. El fin de cadena se marca con CTRL-Z.

Tn Expande los espacios de tabulación a cada n-ésima columna.

U Cambia a mayúsculas todas las letras minúsculas.

V Verifica, en caso de traslado sobre disco, que la operación se haya llevado a cabo correctamente.

W Permite escribir sobre ficheros $ R/O.

Z Pone a cero el bit 7 de los caracteres ASCII en la entrada.

APENDICE G

TABLA DE CONVERSION HEXADECIMAL-DECIMAL

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	00	000
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	0
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	80	31	256	4096
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	512	8192
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	768	12288
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	1024	16384
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	1280	20480
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	1536	24576
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	1792	28672
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	2048	32768
9	144	145	146	147	148	149	150	151	152	153	154	154	156	157	158	159	2304	36864
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	2560	40960
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	2816	45056
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	3072	49152
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	3328	53248
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	3584	57344
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	3840	61440

BIBLIOGRAFIA

Sistemas operativos.
Brinch Jansen. *Escuela de Informática de la Universidad Politécnica de Madrid.*

Sistemas operativos de la computación.
Davis, 1985. *Fondo Educativo Interamericano.*

Sistemas operativos.
Varios autores, 1985. *CEDED.*

Sistema operativo EP/M. Guía del usuario.
T. Hogan, 1983. *Ed. McGraw-Hill.*

Sistema operativo MS-DOS. Guía del usuario.
P. Hoffman / T. Nicoloff, 1985. *Ed. McGraw-Hill.*

Sistema operativo Unix. Guía del usuario.
R. Thomas / J. Yates, 1985. *Ed. McGraw-Hill.*

MS-DOS paso a paso.
Pinaud. *ELISA.*

CP/M palabra a palabra.
Dargery. *ELISA.*

BIBLIOTECA BASICA INFORMATICA

INDICE GENERAL

NOTA: **Ingelek, S. A. se reserva el derecho de modificar, sin previo aviso, el orden, título o contenido de cualquier volumen de la colección.**

NOTAS

Un ordenador no es una unidad simple y autosuficiente, sino que está constituido como un sistema interrelacionado de dispositivos físicos y programas.

Si en el corazón de su ordenador hay un microprocesador de 8 bits existen muchísimas posibilidades de que el sistema operativo de disco de que disponga sea el CP/M. Con su ayuda usted podrá realizar de una forma sencilla y lógica el control de las actividades e interrelaciones de todos los elementos de su sistema sin tener que preocuparse por estudiar hasta el más mínimo detalle de su funcionamiento o de su configuración física.

Ahora bien, ¿quién le puede ayudar en la tarea de aprender cómo aprovechar y usar el CP/M? La respuesta la tiene usted en sus manos.

395 pts.
(incluido IVA)